

**FREE!**  
WITH CU AMIGA

PART FIVE • MAY 1993



FIRST STEPS P10

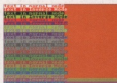
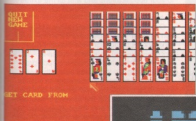
# AMIGA

THE **COMPLETE** GUIDE TO THE AMIGA

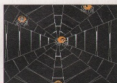
## Guide

### AMOS SPECIAL

THE DEFINITIVE GUIDE TO AMOS



TEXT HANDLING P12



SPRITES P18



SCREEN CONTROL P20



AMOS COMPILER P24

GET THE MOST OUT OF **AMOS**  
IN OUR STEP-BY-STEP GUIDE

**32**  
PAGE SPECIAL

**FREE!**

EXCLUSIVE TO **CU AMIGA** MAGAZINE!  
PART FIVE OF THE MOST COMPREHENSIVE  
GUIDE TO THE AMIGA EVER PUBLISHED.

# CONTENTS

**No doubt you've already been experimenting with last month's fabulous coverdisk giveaway, AMOS. In this supplement, we show you exactly what the program is capable of and how to get the best out of it.**

## Editorial

Welcome to the fifth in the series of CU Amiga supplements. As you have already noticed, this one is dedicated to the superb Amiga programming utility AMOS, given away on last month's coverdisk (if you don't have it, contact the back issues department immediately on 0858 410510) and aims to show you how to get the most from it.

Everyone has wanted to write their own games and utilities at some point. Perhaps you remember playing *Le Mans* on your old Commodore 64 and would love to play it again, only can't find a version of it anywhere on PD. Or you might want an address book/database but can never find one to suit your needs. Until now, the only options have been to wait for the right package to appear or get the next best thing. With AMOS, however, you'll be able to create the package you want, to your specifications and with all the extras and bonuses you need. Sounds too good to be true? Not at all.

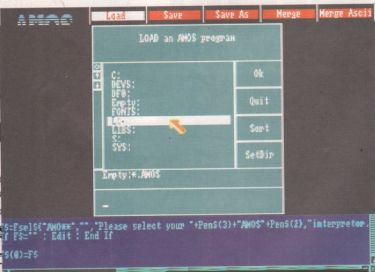
One thing to bear in mind is that AMOS can't do everything. Because it's a high level programming language (see pages 4-5 for more information), programs written in AMOS will never run as fast as machine code programs. Being the thoughtful bunch that we are, though, we were kind enough to include the AMOS compiler on last month's disk too, which helps speed up the running of your self-written software no end. Page 24 explains in more detail how the compiler works and what it can do for you.

If you're a beginner to programming, or the Amiga in general, don't break into a sweat at the sight of all the listings and technical terms used in this guide. AMOS is the perfect tool for the beginner, giving you enough power to create some seriously impressive stuff, while at the same time keeping things simple enough for you to understand what you're doing.

In this guide, we'll take you through the basics of good game design, including a checklist of things to do before you start programming. We'll look at program construction and some of the data handling commands needed by every program. Once you've worked through those two, we can start doing some really interesting stuff, including AMOS's powerful graphic and sound commands.

You'll learn how to add control routines to your programs, and how to achieve a whole host of unusual effects. And if all that isn't enough, we'll even show you how to turn your programs into standalone files, ready to be released onto the PD market, or even onto a CU coverdisk if they're good enough.

Programming is no longer purely the domain of the highly qualified expert. It's an exciting hobby that can often throw more challenges your way than *Monkey Island 2*. No doubt you can't wait to start, so let's go!



The AMOS system is very similar to most other Workbench-friendly utilities, so you should have no trouble at all getting to grips with it.



# Contents

AMOS					
Find	Find Next	Find Top	Replace	Replace All	
Go to	Open	Close	Set Text	Set Tab	
1-4	C-4	Ctrl-Shift	Ctrl-Shift	Ctrl-Shift	Ctrl-Shift

All the important instructions can be found in the menu bars at the top of the screen. See page 8 for more details.

## 4 INTRODUCTION TO AMOS

What's AMOS all about? How does it work? What does it do? How do I use it? All is revealed on these pages.

## 6 THE MENU BARS

The menu bars hold 40 different options. Do you really need that many? We show you what they all do.

## 8 DESIGN

Before you start programming anything, you've got to figure out what you want to program. CU's checklist shows you how to create a game design.

## 10 YOUR FIRST STEP

Within 10 minutes you too can be writing some fairly impressive stuff, using AMOS BASIC commands.

## 12 THE WRITING ON THE WALL

Correct text handling is the first step to professionalism. Lettering in a variety of colours and styles is in your grasp, thanks to AMOS.

## 14 INTERACTION

Mouse, joystick and keyboard control can all be incorporated in your programs with the minimum of fuss, thanks to some very simple instructions.

## 16 REMEMBER YOUR LINES

Lines, boxes, windows and circles can all be used to great effect if you want to create an intuition style interface. You won't believe how easy it all is.

## 18 SPRITES

One of the best things about the Amiga is its sprite handling capabilities. AMOS makes full use of them.

## 20 SCREENS

You can have your own eight-way scrolling backdrop, using a handful of commands – find out how here.

## 22 SOUND

What your program needs is a really jazzy soundtrack, or some effective spot effects. We show you how to stop your programs being aurally challenged.

## 24 COMPILER

So you've finished your program, and you want to release it into the public domain. This page shows you how.

## 25 WHERE NOW?

AMOS is a very expandable package, and there's a lot more to it than meets the eye. If you want to know more, or fancy upgrading, then here's a look at the next step.

## 27 TOTALLY AMOS

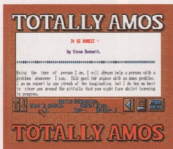
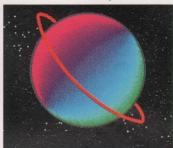
We take a look at one husband and wife team who have turned their interest in AMOS into a much-needed service.

## 28 GOING PUBLIC

AMOS has been put to good use in an amazing variety of PD games. Just to show you what the package really is capable of, we take a look at the best of the bunch.

## 30 AMOS USER GROUPS

Want to get in touch with other AMOS users? Here's a rundown of groups worldwide – and your chance to win a pile of software.



### EDITOR

Tony Dillon

### COMPUTER GRAPHICS

Paul Fleet

### WRITTEN BY

Tony Dillon

### PUBLISHER

Garry Williams

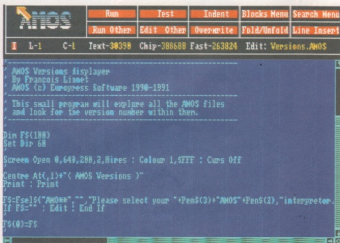
AMIGA GUIDE  
CU AMIGA  
EMAP IMAGES  
30-32 FARRINGTON  
LANE  
LONDON EC1R 3AU

This issue of the Amiga Guide is free with the May 1993 issue of CU Amiga and must not be sold separately.

© 1993 EMAP Images. All rights reserved. No part of this publication may be reproduced in any form without prior permission from the publisher.

# INTRODUCTION

**Now that you've got AMOS loaded, you're probably wondering exactly what it is and what it's capable of. Here's the place to find out...**



The AMOS editor screen. Get used to it, because you'll be seeing a lot of it in the future!

AMOS is intended first and foremost as a game creation package, but it can be used for far more. Stepping around the usual problems associated with coding (tedious routine writing, masses of indecipherable codes and figures) AMOS uses an English-based parser to give novices the tool to create anything they want. A gateway to your Amiga's power, if you like.

In the past, game creators have generally been extremely limiting. You can spot a game

written in *The Quill* and *3D Construction Kit* a mile off. Some say you can spot a game written in AMOS a mile off, but this is untrue. With AMOS, you can write anything you want, with only your imagination and experience shaping any barriers. To begin with, you probably won't be able to make *Midwinter 4*, or *Elite 3*, but don't let that stop you from trying.

## PHRASE BOOK

AMOS itself is a compilation of two programming languages, the AMOS BASIC interpreter and the AMAL command language. The interpreter handles almost all of your code, turning your English commands into machine language for the processor to run. AMAL is the sprite animation suite that handles all the sprite and Bob (Biliter Object) routines. Together, they are capable of some fairly amazing things. But before you can use them, you need to understand how they work.

The interpreter is a derivative of BASIC (Beginners All-purpose Symbolic Instruction Code), the age-old language favoured by both schools and novices. Rather than try and work the processor directly, the interpreter provides you with a suite of over 500 English commands, which make for readable listings and a good understanding of what things do right

## THE FILE SELECTOR



The file selector can look a little daunting at first, but don't worry, it's easier than it looks.

It's probably worth your while getting to know the file selector, as it's something you'll be seeing a lot of. It works in much the same way as any other, with the slider bar on the left sliding through the list of files, and the standard 'OK', 'Sort' and 'Quit' buttons on the right. But how do you actually use it?

First, click on the 'Load' button on the main menu bar. The file selector appears, and after a moment the list of files appears. If the file that you want is listed, fine, just double-click on its name to load it.

If, however, the file you want is in a directory, then you need to single-click on the directory name to open it, and then double-click on the name of the required file.

If your file is on a completely different disk, then remove the disk in the drive and insert the disk with your file on. Click on the small button above the slider arrows with the right mouse button to get a list of devices, and then click on the name of the newly inserted disk.

## TABLE 1

PRINT "What is your name?"

INPUT Answer\$

IF Answer\$="Dan Slingsby" THEN PRINT "Hello Dan, Fancy meeting you here."

IF Answer\$<>"Dan Slingsby" THEN PRINT "Have you seen Dan anywhere?"

# INTRO

1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1302  
1303  
1316  
1309  
1319  
ASLU

1323  
1276  
1277  
1279  
1280  
1281  
1282  
1283  
1286  
1287  
1304  
1307  
1311  
1313  
1324  
1324.2  
1284  
1285  
1299

Geniu

AN

£37.5

AMIGA

57

Ac

Ac

Ac

Lo

Amstrad

Amstrad

Citizen

Citizen

Compaq

Epson

Epson

Epson

NEC

Parsons

Parsons

Ring

054

054



# TO AMOS

from the start. The listing shown in Table 1 is a good example. You can probably already tell what that program will do when you run it.

## THE EDIT SCREEN

Load up AMOS as shown last month (page 14), and take a good look at the main screen, known as the Edit Screen. This is where all the hard work happens. The strip at the top is the menu bar, and we'll be looking at that in just a moment. Below that is the information line, which tells you various things about your system at a glance:

**I (Or O):** Whether the editor is in Insert or Overwrite mode.

**L=1:** Current line

**C=1:** Current column

**Text=:** The amount of memory assigned to the editor

**Chip=:** The amount of chip memory free

**Fast=:** The amount of fast memory free

**Edit=:** The name of the current program

Along the side and bottom of the screen are the scroll bars, which allow you to move quickly and easily around your listing. These are used in exactly the same way as Workbench scroll bars. If you find them too fiddly, you can also move around using the cursor keys, do not fret.

## DIRECT MODE

If you press the escape key, a completely new work screen will appear. This is called Direct Mode, and it acts on each command as you type it, rather than waiting for you to run the program. If you typed PRINT 12/17 in Edit mode, nothing would happen until you ran the program. If, however, you type it here, the command is executed immediately without affecting the listing in Edit mode.

Direct mode allows you to try out commands before they form part of your program, as well as carry out various house-keeping duties without disturbing the flow of your programming. If you wanted to see how many sprites or samples you had in memory, check how much disk space was available or see how two colours went together, this is the place to do it.

## RUNNING A PROGRAM

To load a program, you need to click on the 'Load' option in the menu bar, and then choose the file using the selector (see panel). Once it has loaded you'll be presented with the complete listing. Now, to run it, all you need to do is press F1, or click on 'Run' in the menu bar.

To stop a program in its tracks, without waiting for the logical end, you need to hold down the Control key and press the C key at the same time. This aborts the current program and returns you to the edit screen. To see what I mean, load the 'Scrolling Text Demo' from your AMOS program disk, run it and then abort it.

## MEMORY BANKS

AMOS is capable of some fairly nifty sprite and sample handling, but like any other program the data for these need to be in memory at all times, and saved with the basic program.

This is done by using the AMOS Memory banks, 15 blocks of RAM used specifically for resource data. Once something is loaded into a memory block, it is automatically saved with the program, so there's no need to reload any of it the next time you load. To see how it all works, go to Direct mode, and load the sprite file on this month's coverdisk by typing:

**LOAD "Kittens.Ask"**

Once the file has loaded, jump back to edit mode. Notice how there's no listing? So how do you check if AMOS has loaded anything? Simple. Go back to direct mode and type:

### LISTBANK

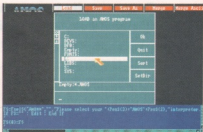
A list of the currently used memory banks appears, and there you should see bank 1 contains five sprites. To see them in action, enter:

**Sprite 1, 200, 200, 1**

We'll deal with sprites in more detail later on in this booklet (page 18).



AMOS direct mode allows you to try out commands and do a little housekeeping without disturbing your programs.



## VARIABLES

One of the most important things to know about when using AMOS is variables. A variable is a named area of memory used for storing information, such as a name, a number or any string of characters. In the case of the listing on this page, **Answers** is the name of the variable. Think of it as a pigeon hole called **Answers**, and whenever the computer looks back at that hole, it will be able to read your answer. A variable can be called almost anything you like, as long as you follow these rules.

- A name by itself means an integer variable (no decimal places). A, Answer and Total can all be used as integer variable names
- A name with a # after it is a real number variable. The command **A=10/4** would give the result 2.5
- A name with a \$ after it is a string variable, meaning that the variable will only have characters.
- A variable can't have the same name as an AMOS command. PRINT, NEXT and DIM can't be variable names, but PRT, NXT and D can.
- Variables are reset every time the program is started, so don't expect to see the same data the next time you load your file.
- A variable will only work in the part of the program it belongs to. To make a variable available for the entire program, including procedures, you need to make it Global. At the start of your program, include the command **GLOBAL**, and then the names of all the variables you want to use, separated by commas. For example:

**GLOBAL Answers\$, A, Hello!**

# THE MENU BARS

**Do you really know what 'Block Hide' and 'Close All' do? If you're still a little confused by the mysterious menu bars, then read on.**

Those 10 words that you can see at the top of the screen constitute the menu bar, and that little box is going to make your programming life much easier, once you've got the hang of it. It contains 40 useful commands that let you do all sorts of system management tasks without touching a key. To use each one, all you need to do is move the mouse pointer so that it highlights the option you want to select, and then single-click on it with the left mouse button. Alternatively, you could just press one of the function keys. The top five options are selected using the keys F1 to F5, and the bottom row are selected using the keys F6 to F10.

## THE DEFAULT MENU



This menu deals directly with the AMOS editor, and is on screen by default.

This is the menu that is on screen when the package loads, and when no keys are being pressed. It gives you access to two other menus, as well as giving you complete control over the editor. The commands are as follows:

**F1: RUN:** An obvious one really, this option runs the program currently displayed on screen. Before it runs it, it will test it for typing errors and similar bugs. If it finds any, it will alert you and abort the running.

**F2: TEST:** Like the Run option, this one checks the program for errors, alerting you as it finds them. As soon as it finds one, it stops the test and places the cursor next to the error.

**F3: INDENT:** To make your programs more readable, you might want to indent loops and procedures, making them easier to spot when scanning over the listing. Choosing this option automatically indents the program in memory.

**F4: BLOCKS MENU:** This option calls up the blocks menu, which we'll look at later.

**F5: SEARCH MENU:** Another menu that can be called from the default one. Again, read all the way.

**F6: RUN OTHER:** AMOS allows you to hold two programs in memory at the same time. To run the other one, for example a sprite editor, use this option.

**F7: EDIT OTHER:** This option simply switches over between the currently displayed listing and any others that you might have stored in memory.

**F8: OVERWRITE:** This switches between the two editing modes. 'Insert' automatically makes room in the listing for anything you type, whereas 'Overwrite' writes over the current listing, replacing existing text with the new characters.

**F9: FOLD/UNFOLD:** This is used to hide procedures. If you have a particularly lengthy procedure which you find is slowing down your editing of the program, placing the cursor within it and pressing this key 'folds' it into memory, leaving only the title line of the procedure on display. To get your procedure back again, all you need to do is select this option again.

**F10: LINE INSERT:** This option creates a blank horizontal line at the current cursor position, making space for new lines.

## THE SYSTEM MENU



The System menu gives you access to the floppy, as well as use of any accessory programs you may have loaded.

The System menu gives you access to the disk drive, and is displayed by holding down the shift key. With the shift key held, the function keys work as before.

**F1: LOAD:** Again, this one is self-explanatory: it loads a file from disk. You can then select the file using the file selector.

**F2: SAVE:** The opposite of load. Saves the current file to disk.

**F3: SAVE AS:** Lets you save the current file under a different name.

**F4: MERGE:** Inserts a program at the current cursor position without erasing the previous listing. With this, programs can be written in modules and then added together at the end.

# MENU BARS

1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1302  
1303  
1316  
1309  
1319  
ASIU

1323  
1276  
1277  
1279  
1280  
1281  
1282  
1283  
1296  
1297  
1304  
1307  
1311  
1313  
1320  
1324  
1284  
1285  
1299

Genu

AM

£37.5

AM

57

AM

Ac

Lo

Amstrad  
Autostar  
Cassini  
Citizen  
Cemco  
Epson  
Epson L  
Epson P  
Epson  
NEC P  
Panasonic  
Panasonic

Rings

054

054

**F5: MERGE ASCII:** If you like, you can write your AMOS listings using your favourite word processor, remembering to keep the line format the same. Save your document as an ASCII file, and then use this option to load it into the interpreter.

**F6: ACC NEW/LOAD:** Clears all the current accessories from memory, and loads all files off disk that have the 'ACC' extension.

**F7: LOAD OTHER:** This loads another program from disk and puts it in memory without displaying the listing. This is particularly useful for accessories such as the sprite designer, which it is always handy to have stored in memory.

**F8: NEW OTHERS:** Clears all accessories from memory. For accessories, read 'Programs not displayed in the edit window'.

**F9: NEW:** Clears the current program from memory. If the program isn't saved, the interpreter will ask you if you want to save it. Type 'Y' or 'N' to answer.

**F10: QUIT:** Exits AMOS and returns to the CLI. You will be prompted to save your program before the system exits.

## ALTERNATIVE KEY SHORTCUTS

The AMOS edit window features a number of other keyboard shortcuts for menu selection. Here's the full list.

Amiga+L	Load a program
Amiga+S	Save a program
Shift+Amiga+S	Save As
Control+B	Block Start
Control+E	Block End
Control+C	Block Cut
Control+P	Block Paste
Control+M	Block Move
Control+S	Block Store
Control+H	Block Hide
Control+F	Find
Control+N	Find Next
Control+R	Replace
Control+TAB	Set Tab

## THE BLOCKS MENU



The Blocks menu lets you manipulate large chunks of your program with ease, which is useful for tidying things up afterwards.

If you've ever used a word processor, you'll already be familiar with the principle behind 'Cut and Paste'. The Blocks menu lets you lift large sections from your listings and move them around using only a couple of mouse clicks. To show the blocks menu, hold down the Control key.

**F1: BLOCK START:** Marks the start of the block you want to highlight. Move the cursor from the first character and select this option.

**F2: BLOCK CUT:** Removes the highlighted block from the listing and stores it in memory.

**F3 BLOCK MOVE:** Moves the highlighted block to the new cursor position and deletes it from the old position.

**F4: BLOCK HIDE:** Deselects a selected block.



Once a block of text has been highlighted, it can be lifted or removed at the touch of a button.

**F5: SAVE ASCII:** Saves the selected block as an ASCII file, which can then be loaded into any standard word processor.

**F6: BLOCK END:** Marks the end of the block. Move the cursor to the end of the block that you want to highlight and then select this.

**F7: BLOCK PASTE:** Places a block stored in memory at the current cursor position.

**F8: BLOCK STORE:** Copies the block into memory but doesn't affect the listing.

**F9: BLOCK SAVE:** Saves the currently selected block to disk as an AMOS program file. The block can then be merged into another program.

## THE SEARCH MENU



When you select the Find option from the search menu, this prompt asks you for the string to search for.



Where is that text you need? The search menu will look for it for you.

The search menu does exactly what you would expect it to do. It hunts through your listing for a pre-set string of characters, such as a variable name, and then does one of a number of things. To display the Search menu, hold down the Alt key.

**F1: FIND:** This option prompts you for a string of characters, and then searches down from the current cursor position until it finds a perfect match.

**F2: FIND NEXT:** The Find option stops when it finds the first match. This option prompts it to look for another match further along.

**F3: FIND TOP:** This is exactly the same as the Find option, only this one searches from the top of the listing regardless of the current cursor position.

**F4: REPLACE:** Searches through your listing for a match, and then replaces it with a second string input at the start of the search. If you have a variable name that you want to change, using this option is the easiest way of doing it. You will be asked to confirm each replacement.

**F5: REPLACE ALL:** Changes all copies of a word in your listing.

**F6: LOW<->UP:** Represents case sensitivity. In this mode, the search routine differentiates between upper and lower case characters. Clicking on this changes the mode to LOW=UP, in which upper and lower case letters are treated as identical.

**F7: OPEN ALL:** Opens all closed procedures in your program.

**F8: CLOSE ALL:** Closes all open procedures in your program.

**F9: SET TEXT B:** Lets you change the size of memory available for your listings. The more memory you have, the larger the programs you can fit into memory.

**F10: SET TAB:** This allows you to set the number of character spaces between each tabulation marker.

# GAME DESIGN

**Design is the backbone of any program. With a good design, the whole programming experience becomes much easier to handle. Here's what to look for.**

Before any programmer can put his or her fingers to the keyboard, they must put pen to paper. A solid game design is the key to successful programming, as anyone who has tried to work without one will no doubt tell you. By figuring out right at the beginning how your program will work, what it will do and what it will look like, you'll save yourself all sorts of hassle later on in the project.

But, you might be asking, how exactly do I make a design? The first thing you need to do is work out exactly what you want to create, and then sketch a couple of notes. If, for example, you wanted to create a shoot 'em up, then you might write something like: "Big Guns will scroll both ways over a dozen levels, each one set on a different planet. The aim will be to shoot a certain number of alien eggs before they spawn alien ships. There will be four different weapon upgrades, ranging from a simple laser to homing missiles. At the end of each level, you'll be able to buy the weapons with your points - the higher your score, the better the weapons you can buy."

## ON YOUR OWN TERMS

OK, so that's your brief. Now you need to think about that in programming terms. How will you make the game scroll both ways? How can you tell when the eggs are ready to gestate? How are the different weapons going to be represented? All these decisions must be made early on, just for the sake of practicality. Some things might not be possible without a loss of speed or playability, and correct plan-

ning will ensure that you spot these things early on.

## RUNNING ORDER

Now comes the tricky part. You have to work out a running order for the program listing itself and begin to formulate the routines and patterns involved in making your program work the way you want it to. How are you going to get the ship and the backdrop scrolling? How many times a second should you read the joystick? That sort of thing. It might seem daunting, so let's break it down into smaller programs.

Figure out the various components of your game, and work on each separately. In the case of the scrolling, your notes would look something like this:

(Variable D=Scrolling direction (1=left, 2=right)  
display screen  
check D  
adjust screen position accordingly  
display screen again

This may not sound much like a guideline for a game, but it's only preparation for a flowchart.

## FLOWING NICELY

As a rule, flowcharts are impossibly dull to create. However, they are also invaluable when writing a game. If you are creating an especially long listing, it's handy to have a list to refer to in order to stop you from getting lost.

A flowchart breaks a program down into single steps, making the entire challenge much easier to cope with. In a game like the one that we have referred to above, writing an engine that moved the backdrop and kept track of all the sprites on screen at once from scratch would be too much for the beginner. Working from a well-written flowchart makes it simple enough for even a Megadrive owner to get to grips with.

To design your flowchart, you need to break your program down again, into the smallest jumps you can. Remember, the more you break it down now, the less you have to figure out later on.

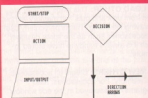
## OTHER POINTS TO CONSIDER

The listing isn't the only thing that needs a lot of thought put into it. You should also

spend some time working out how the game will look and sound. Can you draw well? Are you able to compose a suitable soundtrack for the game?

It's best to be honest with yourself - if you don't think you're up to a certain task, then find someone who is. It makes all the difference in the end.

## FLOWCHART SYMBOLS



Although you don't have to follow the old-school flowchart style, it can make things a lot easier, for you in the long run. Here are the five main symbols you will find yourself working with.

### 1) ACTION

This represents anything done within the computer, such as calculating variables, reading data or setting up a screen.

### 2) INPUT/OUTPUT

This represents any form of input or output, from displaying something to reading the joystick.

### 3) START/STOP

This one is obvious - I don't really need to tell you what it is for.

### 4) DECISION

For these moments when the program can go one of two or more ways, a decision box shows the possible routes. Write the question in the box, and label the exit arrows where possible.

### 5) ARROWS

The direction arrows show the flow of the program, and it's vitally important that you mark the direction on them; without them, routines such as procedures become impossible to understand.

# GAME DESIGN



# FIVE-STAR

QUALITY SOFTWARE (0827 68496)

90p PER DISK

### EDUCATION

1001 ALPHABET  
1002 BASIC READER  
1003 BASIC READER  
1004 BASIC READER  
1005 BASIC READER  
1006 BASIC READER  
1007 BASIC READER  
1008 BASIC READER  
1009 BASIC READER  
1010 BASIC READER

### DEMOS

1011 DEMO 1  
1012 DEMO 2  
1013 DEMO 3  
1014 DEMO 4  
1015 DEMO 5  
1016 DEMO 6  
1017 DEMO 7  
1018 DEMO 8  
1019 DEMO 9  
1020 DEMO 10

### AMIGAS

1021 AMIGA 1  
1022 AMIGA 2  
1023 AMIGA 3  
1024 AMIGA 4  
1025 AMIGA 5  
1026 AMIGA 6  
1027 AMIGA 7  
1028 AMIGA 8  
1029 AMIGA 9  
1030 AMIGA 10

### CLIPARTS

1031 CLIPART 1  
1032 CLIPART 2  
1033 CLIPART 3  
1034 CLIPART 4  
1035 CLIPART 5  
1036 CLIPART 6  
1037 CLIPART 7  
1038 CLIPART 8  
1039 CLIPART 9  
1040 CLIPART 10

### CLIPARTS

1041 CLIPART 1  
1042 CLIPART 2  
1043 CLIPART 3  
1044 CLIPART 4  
1045 CLIPART 5  
1046 CLIPART 6  
1047 CLIPART 7  
1048 CLIPART 8  
1049 CLIPART 9  
1050 CLIPART 10

### EDUCATION

1051 ALPHABET  
1052 BASIC READER  
1053 BASIC READER  
1054 BASIC READER  
1055 BASIC READER  
1056 BASIC READER  
1057 BASIC READER  
1058 BASIC READER  
1059 BASIC READER  
1060 BASIC READER

### DEMOS

1061 DEMO 1  
1062 DEMO 2  
1063 DEMO 3  
1064 DEMO 4  
1065 DEMO 5  
1066 DEMO 6  
1067 DEMO 7  
1068 DEMO 8  
1069 DEMO 9  
1070 DEMO 10

### AMIGAS

1071 AMIGA 1  
1072 AMIGA 2  
1073 AMIGA 3  
1074 AMIGA 4  
1075 AMIGA 5  
1076 AMIGA 6  
1077 AMIGA 7  
1078 AMIGA 8  
1079 AMIGA 9  
1080 AMIGA 10

### CLIPARTS

1081 CLIPART 1  
1082 CLIPART 2  
1083 CLIPART 3  
1084 CLIPART 4  
1085 CLIPART 5  
1086 CLIPART 6  
1087 CLIPART 7  
1088 CLIPART 8  
1089 CLIPART 9  
1090 CLIPART 10

### CLIPARTS

1091 CLIPART 1  
1092 CLIPART 2  
1093 CLIPART 3  
1094 CLIPART 4  
1095 CLIPART 5  
1096 CLIPART 6  
1097 CLIPART 7  
1098 CLIPART 8  
1099 CLIPART 9  
1100 CLIPART 10

### GAMES

1101 GAME 1  
1102 GAME 2  
1103 GAME 3  
1104 GAME 4  
1105 GAME 5  
1106 GAME 6  
1107 GAME 7  
1108 GAME 8  
1109 GAME 9  
1110 GAME 10

### ANIMATIONS

1111 ANIM 1  
1112 ANIM 2  
1113 ANIM 3  
1114 ANIM 4  
1115 ANIM 5  
1116 ANIM 6  
1117 ANIM 7  
1118 ANIM 8  
1119 ANIM 9  
1120 ANIM 10

### UTILITIES

1121 UTIL 1  
1122 UTIL 2  
1123 UTIL 3  
1124 UTIL 4  
1125 UTIL 5  
1126 UTIL 6  
1127 UTIL 7  
1128 UTIL 8  
1129 UTIL 9  
1130 UTIL 10

### GAMES

1131 GAME 1  
1132 GAME 2  
1133 GAME 3  
1134 GAME 4  
1135 GAME 5  
1136 GAME 6  
1137 GAME 7  
1138 GAME 8  
1139 GAME 9  
1140 GAME 10

### ANIMATIONS

1141 ANIM 1  
1142 ANIM 2  
1143 ANIM 3  
1144 ANIM 4  
1145 ANIM 5  
1146 ANIM 6  
1147 ANIM 7  
1148 ANIM 8  
1149 ANIM 9  
1150 ANIM 10

### UTILITIES

1151 UTIL 1  
1152 UTIL 2  
1153 UTIL 3  
1154 UTIL 4  
1155 UTIL 5  
1156 UTIL 6  
1157 UTIL 7  
1158 UTIL 8  
1159 UTIL 9  
1160 UTIL 10

WE ALSO STOCK ASSASSIN GAME (1-40) LSD CHEATS (1-40) FRED-FISH, TBAG AND MUCH MORE

"a flexible and well presented service" ....Mr J.K. from Cotteridge.  
 "that what Amiga DTP users have been waiting for"....Mr D.T. from Leeds.  
 "thankyou for the efficient service"....Mr S.T. from Sheffield.  
 "exactly the service I was looking for"....Mr A.D. from Thetford  
 "thankyou for your excellent service"....Mr D.T. from London E11

## SELECTAFONT COMPANY

Over 1200 Scalable Fonts available for the Amiga in both Compugraphic Style and Adobe Type. Select from over 60 disks. Each Disk contains 12 Fonts and costs just **£3.00 each.**

### BONUS DISKS

Spent £15: 1 Bonus Disk  
 Spent £20: 2 Bonus Disks  
 Spent £25: 3 Bonus Disks  
 Special Prices for Bulk

Please send a large S.A.E. with 28p Postage for a Print-Out of all Fonts. Also state which Programs you will be using them in.

(DEPT CU), 84 Tharpe Road, Hawkwell, Nr Hockley, Essex. S55 4JT.

Probably the finest Collection of Fonts in the U.K.

### SELECTAFONT NEWS CORNER:

1: 250 NEW CG FONTS direct from the USA  
 2: ALL FONTS now available in BOTH FORMATS  
 3: FINAL COPY 2 (update) COMPATIBLE

## The Official AMOS

Public Domain Library  
 1 Pennydyd Road Penlan Suansea SA5 7EH  
 ☎ /Fax 0792 588156 Run by Anne Tucker

### AMOS PUBLIC DOMAIN

NOW OVER 470 DISKS AVAILABLE  
 ROW & EUROPE £2.25  
 Multi disk programs are charged per disks not per title  
**BUY 10 DISKS - GET ONE FREE!**  
 10% Discount for TA Subscribers  
 SAE FOR MORE DETAILS

ROW & ECC prices include VAT 17.5%  
 ROW prices exclude VAT  
 All overseas orders include extra postage

### TOTALLY AMOS

This is the magazine disk for all Amos users which has been approved by Eurogress Software. Here you can meet with other AMOS enthusiasts to learn with other readers.  
 Each disk contains packed FULL of programs and routines which you can load into AMOS and see just how they work, also we have articles and tutorials on lots of AMOS related topics.

Customers/Postal Orders should be crossed and made payable to: AMOS PD LIBRARY  
 by cheque drawn on a UK bank or by International Money Order  
 STERLING ONLY  
 THANK YOU

Hopefully there is something for everyone in each issue of Totally Amos, so join the TA's and see the world of AMOS Programming

**PRICES: UK - £5.00 per issue OR £15.00 for 3 (6 issues)**  
 Please state at which issue you would like your subs to start from  
**ROW £5.00 per issue OR £18.00 (air-mail) for 6 issues.**  
**BACK ISSUES ALWAYS AVAILABLE**  
 Sample issue available as APD 341 Price £2.00  
**ISSUE 10 AVAILABLE IN MAY**

## Amos in Action

By Anne & Len Tucker  
 Learn how to write a master game like Marvin the martian step by step  
 The book has a coupon to send for a free disk £12.95

**CATALOGUE DISK, updated monthly**  
**£1.00 UK £1.50p Row**

FOR MORE INFORMATION PLEASE SEND STAMPED ADDRESS ENVELOPE TO: ANNE TUCKER, AMOS PD, 1 PENNYDYD RD, PENLAN, SWANSEA, SA5 7EH

NAME: \_\_\_\_\_ ADDRESS: \_\_\_\_\_



# YOUR FIRST STEP

**Now that you understand how to transfer your design into a program flowchart, you're ready to program. Here we go...**

Before you can really piece together an AMOS program, you need to spend a little time familiarising yourself with the fundamental components of the language and its construction. I know it sounds like you need to do a degree course, but believe me it isn't that bad. These pages outline the basics of AMOS construction, which you'll need if you're to turn your flowchart into a fully functioning program.

## VARIABLES

Any piece of information which is stored and used again (a player's name, the number of ships left in a shoot 'em up etc.) is stored in memory and labelled. This is known as a variable, so called because the information can vary but the name remains the same. To assign information to a variable, we use the command Let, like this:

```
Let MAG$ = "CU Amiga"
```

MAG\$ is the name of our variable in this case, and we are filling it with the name 'CU Amiga'. Now we have that information stored, anytime we want to use that name, we can call MAG\$. Here's an example:

```
Let MAG$="CU Amiga"
Print "My favourite magazine is ";MAG$
```

See how it works? You can change the information in a variable to almost anything you like - try it. Change the information, and run the program again. Now add these lines to the start of the program, before all the others.

```
Let MAG$="Homes And Gardens"
```

```
Let MAG$="MCN"
Let MAG$="Creative Cricket"
```

What will happen if you run the program now? Run it and see. The contents of a variable can be altered as many times as you like, but the program will always replace the old information with the new.

When naming a variable, a couple of rules need to be followed. Firstly, no two variables can have the same name. Secondly, a variable can't contain the same letters as a program instruction (Print, Run, Draw). Thirdly, some variables need an extender on the end of the name. MAG\$, for example, is a String variable, denoted by the dollar sign (\$) at the end of the name. You can put anything you like in a string variable, but bear in mind that any numbers you store here are stored as characters rather than mathematical symbols - you won't be able to use them in mathematical terms. All strings are enclosed in quote marks (""); without them you'll get an error message.

There are two other kinds of variable recognised by the system. The first are Integers (Whole numbers). These have no extender after the name, and only numbers can be stored in these. If you try to store a string of characters, you'll get an error message. The other kind are Real numbers, which allow decimal places, unlike Integers. A real number is recognisable as having a hash (#) after the name. With that in mind, can you tell which of these are legal and which aren't? Try them and find out.

```
Ed$="Dan Slingsby"
Age=21
Time=12.50
Time$=12
Run=50
Precise=3.14159
```

## CONTROL ROUTINES

AMOS contains a variety of different commands for controlling the flow of your program, which range from simple directions to condition testing and directing a program depending on the outcome of a variable. These will probably seem a little complicated at first, but try them out a few times, and you'll find them a lot

simpler to use than an equivalent program that doesn't use them!

## FOR...NEXT

If you have a segment of program that needs to be repeated a certain number of times, a simple loop is the easiest way to do it, rather than write out the same piece of code over and over again. If you wanted to print your name 20 times, you could write:

```
Print "My Name"
Print "My Name"
Print "My Name"
```

and so on, but surely it would be far easier to use something like:

```
For A=1 to 20
  Print "My Name"
Next A
```

'A' is a variable, and can be anything you like. See how it works?

## DO...LOOP

If you have a piece of program that you want repeated indefinitely, looping forever, then a Do...Loop loop is all you need. Do marks the start of the loop, and the Loop command tells the program to go back to the Do instruction.

```
A=0
Do
  Print A
A=A+1
Loop
```

## REPEAT...UNTIL

Let's assume with the program above that you want it to count to a present random number. There are two ways to do this - one is to do a For Next loop with a random number in the For instruction. The other is to do a Repeat...Until, where the program will break the loop once a condition has been met. Try this:

```
Z=Rnd(30000)
A=1
Repeat
```

# FIRST STEP



```
Print A
A=A+1
Until A=Z
Print "Found it At Last"
```

### IF...THEN...ELSE

Condition testing is the heart of programming. An If...Then instruction is the heart of decision making - we do it every day. If it's warm THEN don't wear a coat, that sort of thing. In the programming sense, it works in exactly the same way. Try this:

```
z=0: B=Rnd(9)+1
Repeat
Print "Give me a number between 1 and 10"
Until A
If A=B then Print "Correct!":Z=1
If A<B then Print "No, sorry"
Until Z=1
Direct
```

The three condition testing symbols are:

'='	Equal to
'<'	Less Than
'>'	Greater than
'<='	Not equal to

Combinations of these can be used (provided that they don't contradict each other - something can't be equal and not equal!) in any of the condition tests.

If you like, you can extend the instruction to include 'Else'. This tells the machine what to do if the condition isn't true. With this, our new program would look something like:

```
z=0: B=Rnd(9)+1
Repeat
Print "Give me a number between 1 and 10"
Until A
If A=B Then z=1 Else Print "Sorry, try again."
Print "Well done!"
Direct
```

### WHILE...WEND

A While...Wend loop is similar in principle to a Repeat...Until loop in that it waits for a condi-

## TABLE 1

```
Do
  Cls
  Print "1) Option 1"
  Print "2) Option 2"
  Print "3) Option 3"
  Print "4) Goto Editor"
  Print "5) Goto Direct"
  Input A
  On A OP1,OP2,OP3,Edit,Direct
  Loop
Procedure OP1
  Cls
  Print "You chose Option 1"
  Wait Key
End proc
Procedure OP2
  Cls
  Print "You chose Option 2"
  Wait Key
End proc
Procedure OP3
  Cls
  Print "You chose Option 3"
  Wait Key
End proc
```

tion to be met before it breaks the loop. The instruction While is followed by the condition, and Wend signals the end of the loop. For example:

```
X=0
While x<20
  locate x,0
  print ""
  X=X+1
wend
Direct
```

### END/EDIT/DIRECT

These are used to end the program. The first, End, just stops things in their tracks, and asks you which mode to go to. Edit ends the program and goes straight back to Edit mode, and Direct ends the program and goes straight to Direct mode. To see how they work, replace the 'Direct' command at the end of the last program with 'End' or 'Edit'.

### ON...PROC/GOTO

On... is a very powerful command indeed. It works with an integer variable to determine where the program should branch to. An example of this is a menu screen. If you wanted, you could just put:

```
If a=1 then PROC1
If a=2 then PROC2
```

and so on. Or, you could use a command like:

```
On A PROC1, PROC2, PROC3, PROC4...
```

Try the listing in Table 1 to see how it works.



Three diagrams to show how the different control structures work. The Do-Loop routine will run forever, but the For-Next and the Repeat-Until loops will run until a specified condition has been met, such as a score reaching a certain level.

## ARRAYS



Variables can be clustered together in Arrays like this one.

Let's say you are dealing with a large set of variables in the same format - as in a database. You could name each variable separately, (X1, X2, X3, X4, X5), but that wastes a lot of memory and makes the program hard to follow. What you need is an array - a collection of shoe boxes stuck together. An array has a single name, and the contents are called using a co-ordinates system. Arrays can be in as many dimensions as you want, from two to 10 dimensions.

### DIM

Dim creates a new array ready for filling. DIM CU(10) creates an array with 10 spaces. DIM CU (10, 10) creates an array with 10 main spaces broken into 10 zones, giving 100 spaces.

### READ, DATA

Filling an array by hand can be time-consuming, if you use Let array(1)=XXX, Let array(2)=XXX etc. The simplest way to do it is to set up a Read statement and a collection of data. Read reads the next item of data and puts it in a preset position, as shown here:

```
Dim STAFF$(7,2)
For place=1 to 2
  For name=1 to 7
    Read STAFF$(name, place)
  Next name
Next place
Data "Dan", "Editor", "John", "Dep Ed", "Nick", "Tech Ed", "Gordon", "Design", "Mat", "Tech Advice", "Tony H", "Star Fighter", "Tony D", "Freelance"
```

Run the program, and nothing will happen. What you need to do now is check that the array has been filled. Add the following lines to the program:

```
For Name=1 to 7
  Print STAFF$(name,1),STAFF$(name,2)
Next Name
```

Handy, isn't it?





# PUBLIC DOMAIN SOFTWARE AT ITS BEST!

**WE STOCK THE LOT!**  
FISH TO 810!  
AMOS DISKS!  
TBAG DISKS!  
NZ DISKS!  
AMICUS!  
AMIGAN!  
ALL CLR TITLES!

FAST SAME DAY SERVICE, HELPFUL SALES STAFF, ESTABLISHED FOR OVER 5 YEARS  
40,000 MEMBERS THROUGHOUT THE WORLD, WELL OVER 4000 TITLES IN STOCK!

## HOW TO ORDER

BY PHONE  
(0924) 366982  
CREDIT CARD / SWITCH

BY FAX  
(0924) 200943  
WITH ORDER & CREDIT CARD  
DETAILS

BY POST  
PLEASE MAKE CHEQUES  
PAYABLE TO :-  
17 BIT SOFTWARE  
1ST FLOOR OFFICES  
2&8 MARKET STREET  
WAKEFIELD  
WEST YORKSHIRE  
WF1 1DH

OFFICE HOURS  
MON-THURS 9.00 TO 8.00  
FRI & SAT 9.00 TO 5.30  
WE ARE OPEN TO PERSONAL  
CALLERS FROM 9.00 TO 5.30.

## DISK PRICES

17 BIT, FISH ETC.....	£1.25
SCHEME 17.....	£2.00
AM/FM MAG.....	£2.50
AM/FM SAMPLES.....	£2.50
CLR SINGLE TITLE.....	£3.50
CLR 2 DISK SET.....	£4.50
CLR 3 DISK SET.....	£4.99
CAT DISKS.....	50P

## POSTAGE RATES

UK PD ORDERS.....	50P
OVERSEAS ORDERS.....	20%
MIN OVERSEAS P&P.....	1.00p
PLEASE ADD 75P P&P FOR COMMERCIAL GAMES, DISK BOXES ETC.	

BUY 10 DISKS AND GET  
1 EXTRA DISK FREE!  
BUY 20 DISKS AND GET  
3 EXTRA DISKS FREE!

PLEASE NOTE: (A) AFTER A DISK  
MAKES 2 DISKS ETC. PLEASE STATE  
THIS WHEN ORDERING. THANKS

## NEW FOR CDTV!

The 17 Bit Collection  
Over 1600 disks worth  
of the best in public domain  
games & demos Etc. All on a  
double CD! Hundreds of  
Demos, Pics, Games & Utilities for  
only £40.99 including P&P!  
Unbeatable Value!

## Commercial Stash!

Amos 3D.....	£18.99
Amos Compiler.....	£18.99
Captive.....	£12.99
Defender Of The Crown.....	£5.99
Dream Team Compilation.....	£17.99
Harlequin.....	£14.99
Lemmings 2.....	£22.99
Lonheart.....	£21.99
Mac Compilation.....	£12.99
Megalomania.....	£11.99
Myth.....	£14.99
No Second Prize.....	£17.99
Nigel Mansell World Cup.....	£17.99
Puffy.....	£18.99
Realms.....	£11.99
Streightflier II.....	£18.99
Spritz (Art Package).....	£9.99
Thunderblitz.....	£17.99
Wing Commander.....	£16.99
Zool.....	£17.99

## HOT TEAM 17 SOFTWARE!

Allen Breed Remix.....	£9.99
Assassin.....	£18.99
Body Blows.....	£18.99
Project X.....	£17.99
Superfrog.....	£18.99

## LSO "LEGAL TOOLS!"

We currently stock all the LSO legal tools  
utility compilations from 1 to 80! A catalogue is  
available of these titles for £1.00 inc P&P  
or free with orders of 10 disks or more.

## Attention Overseas Traders!

We are currently looking for reliable companies  
to represent us in the overseas market. If you  
would like to be an official 17 Bit Dealer, and  
would like to know how YOU can benefit,  
Contact us by Tel. or Fax NOW!  
Join The Leaders in Public Domain!

## CDPD VOLUME #1

Containing Fred Fish disks from F001  
to F660!  
Well Worth £19.99 + 75p P&P  
That's over 650 disks Worth!

## CDPD VOLUME #2

Continuation of Fred Fish from F661  
To F750 + The Entire SCOPE & JAM  
Ranges! Another packed CD for only  
£19.99 + 75p P&P

## DEMO CD VOLUME #1

For the Connoisseurs, this CD  
contains Demos, Intros, Cliparts,  
Modules, Samples etc!  
Only £19.99 + 75p P&P

## Bits 'N' Bats

### "SPACE WARS"

See the latest in Amiga Animation  
on VHS Video! 24 bit dynamic  
Hi-Res movie from T. Richter  
Only £11.99 + 75p P&P

### "HOBBITS & SPACESHIPS"

Ex Composer Boon Lynne brings you  
the latest in Synth Sounds on Audio  
CD. 10 superb tracks running for  
72 mins for only £12.99 + 75p P&P

### "THE FINAL FRONTIER"

A 4 disk mag which no Trekkie  
should be without. Includes exclusive  
artwork by T. Richter and upto date  
seminar news etc. Only £6.95

### "AM/FM"

Issue 11 of this ever popular disk mag  
for music enthusiasts is now available  
Only £2.50 or £5.00 with sample disk  
Back issues also available!

### "LSO GRAPEVINE #14"

As always, our most popular disk  
mag is packed with controversial  
topics and news from the "Scene".  
Don't miss it at £3.75!

### ASSASSIN'S GAMES DISKS!

If you thought the first 30 were good,  
you should see the next lot! Now a total  
of 56 issues available at incredible  
prices! Have a look below!  
Any 10 for £11.99 Any 20 for £21.99  
Any 30 for £29.99 Any 40 for £37.99  
Any 50 for £45.99 or just take the  
whole lot for an incredible £99.99!

## LATEST DISKS!

+2503	Windblend A1200 FracGen.
+2502	More 1200 Only Utis
+2501	Childrens Songs
+2500	Gladators Music Disk
+2499	Fruit Salad Game
+2498	Picture Puzzle
+2497	Marcos Box Of Fun (1.5MB)
+2496	Excel Database
+2495	Attraction Music Disk
+2494 (AB)	18th Hole Golf Game
+2493	The Enforcer Anim
+2492	Ham-8 Videos
+2491	Vidi Effects
+2490	Drum Loops/Samples
+2489	Cindy Crawford Slides #2
+2488	Cindy Crawford Slides #1
+2487	Tad & Madgads Menus
+2486	Menu Launcher
+2485	PolyED V1.0
+2484	The Money Program
+2483	Cit-Tris & Pacatask
+2482	Hillbilly Anim
+2481	Baghman
+2480	Amy PD Review V1.1
+2479	Singalong Nursery Rhymes
+2478	How The Earth Began
+2477	Flashback Demo
+2476	Bomb Jacky
+2475	Create Adv. Games V2.1

## NEW CLR TITLES!

CLU11	CALC V1.3
CLU12	Virtual Windows V1.0
CLU13	Datos
CLU14	Stock Controller
CLU15	Epoch VI
CLE10 (3)	Basically Amiga
CLE14 (3)	Ecology
CLE15	Fastfret II
CLE16 (2)	King's & Queens
CLE17	Thingamajig
CLE18	Work & Play
CLE19	Play It Safe
CL018	Van 29
CL020	Jungle Bungle
CL021	Flower Power
CL022	Stocking Filters
CL023	Marvin The Martian
CL024	Easy Money

## SCHEME 17!

SS23	Personality Analysis
SS22	Lockout (H.D Security)
SS21	Mindmatch
SS19	Techno Attack IV
SS18	Pools Predictor
SS17 (3)	Finlandia - £2.95
SS16	Freakezone
SS15	Kids Karaoke
SS14	Crystal Symphonies II
SS13	Techno Attack III
SS12	Christmas Karaoke



# INTERACTION

*A game often stands or falls on its control method. On these pages we show you how to ensure that yours is a winner.*

You already know one method of entering data from the keyboard – the input command – but there are a variety of others which can be used to far greater effect.

## INKEY\$

Inkey\$ tests the keyboard to see if a key is being pressed, and enters it directly into a variable. Whereas input requires a 'return', Inkey\$ works immediately. If no key is pressed, the instruction leaves the variable blank and carries on. See Table 1.

**TABLE 1**

```
Do
  X$=Inkey$
  Print x$;
Loop
```

But what if you want the program to wait until you have pressed a key? One way to do this is to stick a small If...Then line in the program. See Table 2.

## SCANCODE

Scancode is used to check the internal number for any of the keys on the Amiga keyboard, including the ones with no visible effect, such as 'Help' or the function keys. See the example in Table 3.

Once you have your Scancodes, you can use them in conjunction with the Key State command, which tests whether a key is currently being pressed. If the Key State test is true, it will return a result of True. To see what I mean, have a look at Table 4.

## INPUT\$

Input\$ is a different command to Input, so read this carefully. Input\$ asks for a set number of characters, and places them in a nominated variable. See Table 5 for an example.

**TABLE 2**

```
Do
  z=0
  Repeat
    x$=Inkey$
    If x$<" " then z=1
  Until Z=1
  Print x$;
Loop
```

**TABLE 3**

```
Do
  While K$=""
    K$=Inkey$
  Wend
  If Asc(k$)=0 Then Print "You
  Pressed A Key With No ASCII Code!"
  Print "The Scancode is ":"Scancode
  K$=""
Loop
```

**TABLE 4**

```
Do
  If Key State (69)=True Then Print
  "You've Escaped!"
  If Key State (95)=True Then Print
  "No I won't help you!"
Loop
```

## WAIT KEY

This instruction pauses the program until any key has been pressed.

## STICK AROUND

Reading the joystick in AMOS is something that you will inevitably be using a great deal, so you'll be happy to know that there are a few simple commands that make this just as easy as printing your name. The commands are as follows:

## JLEFT, JRIGHT, JUP, JDOWN, FIRE

These five commands check if the various directions (counting the fire button as a direction) are being used, returning a value of 1 if the test is true. The number in brackets which follows the instruction is the number of the port under test. See Table 6.

**TABLE 5**

```
Clear Key: Rem Clears keyboard
buffer
Print "Please type 10 letters"
c$=Input$(10): Print "You typed ":"c$
```

**TABLE 6**

```
Do
  If Jup(1) Then Print "Up"
  If Jdown(1) Then Print "Down"
  If Fire(1) Then Print "Fire"
Loop
```

## MOUSE MOUSE

The Amiga is perfectly suited to mouse-controlled games, and AMOS is more than capable of creating those games.

## HIDE/SHOW

These two commands are used to hide and redisplay the mouse pointer, for joystick-controlled games such as shoot 'em ups. Use the command Hide On to remove the pointer, and Show On to redisplay it.

# INTERACTION

1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400

Genu

E37.5

A M I G A

St A

AC

Lo

Amiga

Amiga

Claris

Claris

Comm

Comm

Epson

Epson

Epson

Epson

NEC

Parsons

Parsons

Parsons

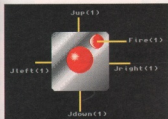
Parsons

Parsons

Parsons

Parsons





The five joystick directions and the commands used to read them. Remember, the number in brackets is the joystick port.

## CHANGE MOUSE

This instruction lets you change the graphic used for the mouse pointer. There are three pointers in memory at all times, so the command Change Mouse 1, 2 or 3 changes the pointer to an arrow, a crosshair or a clock. Any number higher takes a graphic from the sprite bank. Your only limit is that the graphic can't be any more than 16 pixels wide or have more than four colours.

## MOUSE KEY

This checks the status of the mouse buttons, and returns a bit-pattern. To see the bit patterns available, try the program in Table 7.

### TABLE 7

```
Do
  Locate 0,0
  M=Mouse Key: Print "Bit Pattern
  Number ";M
Loop
```

To do a one shot test of the mouse buttons, to see if a button has been 'clicked', use the Mouse Click command instead.

## X MOUSE, Y MOUSE

These two commands fill double functions, depending on the way they are used. In the format 'Variable=X Mouse', the current X hardware co-ordinate (which isn't always the same as the screen co-ordinate) of the mouse is stored in a named variable. This is useful for testing where the mouse is.

By inverting the command, and using it in a different way (X Mouse=100), you can set the X hardware co-ordinate, thereby moving the mouse to a new position. See in Table 8.

## LIMIT MOUSE

Normally the mouse has the run of the screen, but you can limit its movements to a rectangular portion by defining the top left and bottom right corners of the box of hardware co-ordinates. Try this program:

```
Limit Mouse $0,$0 to $300,$200
Wait Key
```

### TABLE 8

```
Do
  X=X Mouse: Y=Y Mouse
  Locate 0,0: Print "X:";X: "Y:";Y
  If Mouse Click Then X Mouse=Rnd(320):Y Mouse=Rnd(200)
Loop
```

## MENUS

Menu bars are something we all take for granted - anyone who has had more than a week with an Amiga knows that holding down the right mouse button makes a line of menu options appear. With that in mind, one of AMOS's strongest points is its ability to build large and complex menus with minimum fuss.

## MENU ON

Turns on the menu bar. Don't bother doing it at the moment, because you haven't defined a menu yet. To do so, you need to use the Menu\$(1) instruction. This works in two ways. The first is to have a single figure within the brackets, which defines a title for the menu bar. Therefore:

```
Menu$(1)="About"
Menu$(2)="Options"
Menu On
6
```

Creates an active menu bar, but with no options. You need to create the options with the second use of the Menu\$(1) instruction. This time you use two or more figures between the bracket, separated by commas. The first figure shows which menu heading the menu option appears under, the second is the order the item appears in, the third (if there is one) puts the option on a side branch menu. Add the lines shown in Table 9 to the program. Now run the program and see how it works.

## CHOICE

The Choice(1) instruction is used to see which menu option you have chosen. The instruction 'Head=Choice' will read the menu heading number into the variable 'Head'. To read the menu option chosen, you need to number the Choice command. Add the lines in Table 10 to your listing to see what it means.

### TABLE 9

```
Menu$(1,1)="About Menus"
Menu$(1,2)="About CU"
Menu$(2,1)="New Game"
Menu$(2,2)="Old Game"
Menu$(2,3)="Quit"
```

### TABLE 10

```
Do
  If Choice and Choice(1)=1 and Choice(2)=1 Then Print "This is a menu option"
  If Choice and choice(1)=1 and choice(2)=2 Then Print "What do you want to know?"
Loop
```

## ON MENU PROC

Instead of writing out a whole string of commands every time you want to read the menu, you can assign a procedure to each of the menu titles using the On Menu Proc instruction in conjunction with the On Menu On command. This system checks the menu bar 50 times a second without any programmed checks by you, so your program can continue as normal. Try the listing in Table 11.

**Note:** Once 'On Menu Proc' has been used, the On Menu On system stops, so remember to put an 'On Menu On' at the end of each procedure.



Hardware co-ordinates refer to the entire screen, not just what's visible, as this diagram shows.

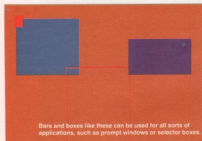
### TABLE 11

```
Menu$(1)="Mouse":
Menu$(2)="Quit"
Menu$(1,1)="Arrow":Menu$(1,2)="
Cross":Menu$(1,3)="Clock"
Menu$(2,1)="Editor":Menu$(2,2)="
Direct"
Menu On
On Menu Proc MSE, QWIT
Rem: Do something
Do
  For x=1 to 100
    print X;
  Next X
Loop
Procedure MSE
  If Choice(2) then Change Mouse
  Choice(2)
  On Menu On
End proc
Procedure QWIT
  If Choice(2)=1 then Edit
  If Choice(2)=2 Then Direct
  On Menu On
End Proc
```

# BASIC GRAPHICS

1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1302  
1303  
1310  
1309  
1319  
ASLU

1323  
1276  
1277  
1279  
1280  
1281  
1282  
1283  
1296  
1297  
1304  
1307  
1311  
1313  
1320  
1324  
1284  
1285  
1299



AMOS has a large collection of tools for defining open and closed polygons and other geometric shapes. All are based on a simple co-ordinate system, with the first figure marking the position across from left (0) to right (320) and the second figure marking the position from the top (0) to the bottom (200 NTSC, 256 PAL).

## RAINBOW BRITE!

Before you draw anything you need to choose your colours. AMOS has a few simple, but effective instructions for palette selection. One thing to note here is the colour index syntax. This is the name given to the settings of the individual colours.

A colour index is a three-figure hexadecimal figure which tells the processor how much red, green and blue should be mixed to create the colour – just like using the colour mixer on a program such as *Deluxe Paint*. Take a look at the right decimal/hex conversion table below.

## INK

The **INK** command is used to set the colour for subsequent drawing operations, and works in exactly the same way as the **PEN** command. See Table 1.

*Before you dive headlong into the slightly difficult world of sprite and blitter objects, why not play around with AMOS's more fundamental graphic tools?*

## TABLE 1

**I=4**

For A = 10 to 100 Step 10

For B = 1 to 10

**INK I**

Draw 10, A+B to 180, A+B

**Next B**

**I=I+1**

**Next A**

colours at once by typing a string of colour indexes separated by a comma. If you don't want to change a particular colour, then just leave a space between that pair of commas.

## PLOT

**PLOT** colours a single pixel on screen using the current ink colour. For example:

**INK 10**

**Plot 100,100**

## DRAW/POLYLINE/POLYGON

The **Draw** command draws a straight line between two points. Both points can be set, or you can leave it to draw from the current cursor position. To see what I mean, go to direct mode, clear the screen using **CLS** and type the following line:

**Draw 100,100 to 200,100**

Bingo, a line appears. Now try the next line:

**Draw to 50,50**

See how the line has automatically been drawn from the end of the last one?

Incidentally, with irregular multiple line shapes, such as polygons, the **Polyline** command works like an extended **Draw** instruction, in that you can stick as many to X,Y's as you like on the end. To draw a filled polygon, use the **Polygon** command. For example:

**Polyline 20,20 to 100, 100, to 80, 150 to 20,140 to 20,20**

draws an empty polygon:

**Polygon 20,20 to 100, 100, to 80, 150 to 20,140 to 20,20**

draws a filled one. Easy!

## BOX/BAR

If you want to draw a hollow box on screen, the easiest way to do it is to use the **Box** instruction. Like the **Draw** command, two sets of co-ordinates are used – these ones specify opposing corners of the box.

## COLOUR

This instruction sets a colour in the palette to a specified colour index. Add these lines to the start of the previous listing, and see if you can guess what colours will appear:

**Colour 4, \$F00**

**Colour 5, \$111**

**Colour 6, \$D00**

**Colour 7, \$123**

**Colour 8, \$AF0**

**Colour 9 \$002**

## PALETTE

Obviously, keying in all those **Colour** commands every time you want to change the palette is going to get very tiresome; with the **Palette** command you can change all the

## DECIMAL/HEX CONVERSION TABLE

**Decimal:** 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

**Hex:** 1 2 3 4 5 6 7 8 9 A B C D E F

To enter hex numbers in AMOS, you need to add the prefix '\$'. For example, to enter the number 15 in hex, you would type \$F. In the colour index, the three digits correspond directly with the red, green and blue settings. Therefore absolute red is \$F00, a medium grey is \$777 and white is \$FFF. Got that?

# BASIC GRAPHICS

Lo

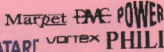
Amstrad  
Amstrad  
Cassini  
Comet  
Epson  
Epson  
Epson  
Epson  
NEC  
Panasonic  
Panasonic

Rings

054



# GASTEINER



Unit 2  
Millmead Business  
Centre  
Millmead Road  
London N17 9QU  
Tel: 081 365 1151  
Fax 081 885 1953

## AMIGA COMPUTERS

2000+	£269.00
2000	£269.00
2000 20Mb Hard Drive	£465.00
2000 60Mb Hard Drive	£549.00
2000	£379.00
2000 20Mb Hard Drive	£529.00
2000 40Mb Hard Drive	£599.00
2000 65Mb Hard Drive	£639.00
2000 80Mb Hard Drive	£679.00
2000 120Mb Hard Drive	£769.00
2000 120 Mb Hard Drive	£2089.00

## MONITORS

Philips CM8833 Mk2	£229.00
Commodore 1084/SSDI	£209.00
Commodore 1960 Multisync	£439.00
Gasteiner Multisync Monitor	£349.00

## PRINTERS

Golden Swift 9 Colour	£179.00
Golden Swift 240 Colour	£279.00
Golden Swift 200 Colour	£219.00
GP 500	£309.00
GP 500 Colour	£439.00
GP 550 Colour	£550.00

## RAM

2000 1/2Mb	£14.95
2000 1/2 Mb With Clock	£19.95
2000 1Mb	£34.95
2000 1Mb With Clock	£39.95
2000 2Mb (PCMCIA)	£119.00
2000 4Mb (PCMCIA)	£189.00
2000 Simms (Gvp)	£27.00
2000 Simms (Gvp)	£89.00
2000 2Mb - 8Mb	£129.00

## SCANNERS

Power Mono	£95.00
Power Colour	£235.00
HiData Mono	£99.00
HiData Plus	£119.95
HiData OCR	£165.00
Gasteiner Mono	£89.00
Gasteiner Scan Read	£129.00
HiScan Pro V3	£89.00
HiScan GT 8000	£1199.00

## HARD DRIVES

### GASTEINER POWER

- \* External IDE HDD for Amiga A500/A500+/A1500/A2000
- \* Memory conveniently expanded to 2/4/6/8Mb by using 1M X 4 Zips
- \* 100% compatible
- \* Easy Installation, Just Plug in and go
- \* Auto boot, Auto config and zero wait states
- Controller for A500/A500+/A1500/ A2000 £99.00
- Controller + 40Mb Hard Drive £249.00
- Controller + 65Mb Hard Drive £279.00
- Controller + 85Mb Hard Drive £299.00
- Controller + 120Mb Hard Drive £329.00

### GASTEINER POWER FOR A600/A1200

- 20Mb + IDE Cable £149.95
- 65Mb + IDE Cable £199.00
- 85Mb + IDE Cable £279.00
- 120Mb + IDE Cable £299.00
- Fitting for A600 or A1200 £29.95

## BARE HARD DRIVES

IDE	SCSI
40Mb £99.00	50Mb £199.00
65Mb £199.00	85Mb £279.00
85Mb £219.00	120Mb £299.00
120Mb £249.00	210Mb £319.00

## ACCESSORIES

- Power Supply A500/A500+/A600/A1200(High Voltage) £34.95
- Power Supply A1500/A2000 £69.95
- Internal Drive for A500 £40.00
- Internal Drive for A2000 £45.00
- A500 Rom Switcher £12.95
- A600/A1200 Rom Switcher £12.95
- Auto sensing joystick/Mouseswitch Box £12.95
- Printer Cable £6.00
- Modem Cable £6.00
- SCSI Cable £6.00
- IDE Cable for A600/A1200 £15.00
- External Drive for A2000 £49.00
- Blitz Amiga £20.00
- ICD Flicker Free Video 2 £199.00
- Commodore 64 Power Supply £19.00
- 10 Blank Disc £7.00
- 3.5" External Drive £50.00

## MICE + TRACKBALL

### AlfaData

- Infra Red Mouse £45.00
- Mega Mouse £10.95
- Mega Mouse (Mat + Holder) £14.95
- 300 DPI Optical Mouse £27.95
- The Trackball £29.95
- Crystal Trackball £34.95
- Optical Pen Mouse £35.95
- Golden Image
- GI-600 £13.95
- Optical Mouse £23.95
- Brush Mouse £19.95
- New Golden Image
- 400 Dpi Mark 2 Mouse £14.95

## SOFTWARE

### Word Processors/Publishing

- Pen Pal V1.4 £49.95
- Final Copy II V2.0 £99.95
- Kindwords 3 £39.95
- Wordworth V1.1 £109.95
- Transwrite £29.95
- Prowrite 3.3 £79.95
- Pagestream V2.2 £129.95
- Professional Page V3.0 £129.95
- Pagesetter II £44.95

### Softclips Clip Art

- Classic Clip Art £29.95
- People Clip Art £29.95
- Collectors Clip Art £29.95
- Animal Clip Art £29.95
- Electric Thesaurus £29.95

### CAD & structuredrawing

- Intro CAD Plus £79.95
- X-CAD 3000 £269.95
- Professional Draw 3 £89.95

### Animation and Graphics

- Deluxe Paint 4 £64.95
- Real 3D Professional Turbo £249.95
- Art Department professional V2 £144.95

### DCTV Composite Video 24 Bit

- graphics System (PAL) £379.95
- Imagine 2.0 £189.95

Phone for access to our massive competitively priced range now!

Products advertised represent a small sample of our instock range. A complete price list is available on request.

**DELIVERY CHARGES** UK MAINLAND (NOT HIGHLANDS)  
Despatched by post please check charges when ordering.  
Next day courier service, 10 per box  
Please enquire  
**IN ADDITION WE OFFER THE FOLLOWING EXPRESS SERVICE**  
Saturday deliveries AM next day.  
Normal rate plus 15% VAT per box  
Normal rate plus 8% VAT per box

E&O.E Price subject to changewithout notice. Goods subject to availability.  
Specifications subject to change without notice. All Trademarks Acknowledged.

# SPRITES

*When was the last time  
you looked at the  
graphics in a game and  
thought, 'I wish I could  
do that'? With AMOS  
you can!*



Attention has to be paid to the backdrop. After all, these  
spiders wouldn't look quite the same if placed...

...on a racetrack! See what I mean?



The Amiga is capable of displaying eight hardware sprites on screen at once. AMOS is capable of displaying up to 64 computer sprites, all kept alive and healthy by the interpreter.

You might think that such a complicated business would require a complicated set of commands, but nothing could be further from the truth. AMOS Basic uses only seven commands to create and use sprites, and then hands over to AMAL to do the rest.

AMAL is the AMOS Animation Language, and is used to create smoothly animating and moving sprites which, once set, can be left to go about their business. To show you how easy it is, we're going to load a sprite and animate it. First, load up a sprite bank - either your own or the 'Spidy.Abk' file on the coverdisk. (Go to direct mode to do this).

Now return to the Editor window, and type the commands:

**SPRITE 8, 200, 100, 1  
DIRECT**



Now run it. That was easy, and getting it moving is just as simple. Enter these lines:

```
a5="Anim 8,(1,0)(2,0);"  
a5-a5="Loop:Move 320,0,100; Move -  
320,0,100; Jump Loop"  
AMAL 8,a5: AMAL On
```

Can you guess what the mysterious AMAL commands are? You'll have to wait until later to see if you're right.

## SPRITE CONTROL

Here are all the AMOS Sprite commands, complete with syntax and examples.

### SPRITE

This command simply creates a sprite and displays it on screen. The instruction is followed by four variables, namely the index number of the sprite, which can be anything between 0

and 63, the X and Y co-ordinates of the sprite and the sprite image which is to be taken from the sprite bank.

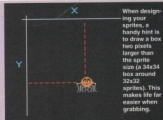
## GET SPRITE PALETTE

It always happens. You have everything set, you load your sprites and when you display them...they look awful. The sprite bank holds the correct colours for the sprites, but unless stated otherwise the sprites take the palette from the current screen, which generally speaking is wrong. So, by sticking this command at the start of your program, you can correct this little problem.

## SPRITE OFF

The Sprite Off command can be used in two ways. On its own, it turns off all sprite activity and removes all sprites from the screen. However, by adding a number to the end of the instruction you can specify a sprite to disable. Try the example in Table 1 (with a sprite bank in memory).

# SPRITES



1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299

1323  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299

Genus

Amiga

£37.50

Amiga

Amiga

Amiga

Amiga

Amiga

Lo

Amiga

Amiga

Amiga

Amiga

Amiga

Amiga

Amiga

Amiga

Amiga



**TABLE 1**

For a=1 to 8  
 sprite a,a\*25,100+a,1  
 next a  
 locate 0,0: Print "Enter number of sprite to disable"  
 Input a  
 sprite off a  
 direct

**SPRITE UPDATE**

Sprite Update is an automatic process that tries to move all sprites during a vertical blank, creating smooth movement. However, if you have a lot of sprites on screen at once, it can't handle them and you end up with some noticeably jerky movements. Use the instruction Sprite Update Off to turn off the automatic process in situations like this, and Sprite Update On to switch it back on.

**X SPRITE, Y SPRITE**

X Sprite and Y Sprite are used to find the X and Y co-ordinates of a nominated sprite - useful when using AMAL Movement commands which don't tell a sprite to stop moving when it reaches the edge. The X Sprite and Y Sprite commands allow you to keep a check on all sprites using the syntax:

variable=X Sprite (Sprite number)

**GET SPRITE**

The Get Sprite instruction does the same job as the sprite grabber in the Sprite Editor program, and allows you to take sprites directly from a screen image. If you know the co-ordinates of the images you want to grab, this is much faster. Load an IFF image to the current screen, and try these commands.

Get Sprite 1,200,100 to 232,132

Get Sprite 2,150,100 to 200,150

Now display the sprites using the Sprite command, and see which areas you've grabbed.

**TABLE 3**

Sprite 8,100,50,1  
 a\$="Move 100,0,50:Move 0,100,50:Move -100,0,50:Move 0,-100,50;"  
 Amal 8,a\$:Amal on

**KEEP ON MOVING**

AMAL has been developed for those people who really don't want to be bogged down with animating and moving sprites by hand, who would rather go without than track every single attack wave in a shoot 'em up. Basically, AMAL has been developed for everyone! It allows you to set movement and animation instructions to a sprite, and then go off and do other things. Load in the 'Spidy Abk' sprite bank, and try the program in Table 2.

**TABLE 2**

Sprite 8,100,100,1  
 a\$="Anim 0,(1,8)(2,8);"  
 a\$=a\$+"Loop:Move 150,0,10; Move -150,0,10; Jump Loop"  
 Amal 8,a\$: Amal on 8  
 do  
 Print "Enter a word"  
 Input z\$  
 Print z\$  
 loop

**AMAL SYNTAX**

AMAL works using string variables - sets of instructions enclosed in quotation marks. Unlike standard AMOS commands, the program doesn't correct case or spacing, so you have to be very careful when entering your AMAL strings. If you entered the program above and got the error message "Error in animation string", check that all the commands start with a capital letter, and that the semi-colons (;) are in the correct spaces.

Once you have created your AMAL string, it has to be assigned to an available sprite with the command AMAL (sprite number).(String variable name), and then switched on with the command AMAL On.

**MOVE**

The most basic of all AMAL commands is the Move instruction. As you might guess, it simply moves a sprite in a certain direction relative to its current position, at a set speed. Note: The co-ordinates you specify in the instruction tell the sprite how far to go, not which co-ordinate to move to. Co-ordinates of 100, 100 will move the sprite 100 pixels to the right and 100 pixels down from its present position. The third variable denotes the number of movement steps allocated. An instruction that moves the sprite 100 pixels using 50 steps will move the sprite two pixels at a time, giving quite smooth movement. Load the sprite bank from the disk and try the example in Table 3.

**ANIM**

Anim animates a sprite at a set speed through a pre-determined series of frames. To tell the program how to animate, pairs of numbers need to be entered into your string to tell the program which frames to display and for how long. Load the 'Spidy Abk' sprite bank and try the example in Table 4.

**TABLE 4**

Sprite 8,100,100,1  
 a\$="Anim 0,(1,8)(2,8)(3,8)(4,8);"  
 Amal 8,a\$:Amal on

Experiment with different speeds and orders to see how it works.

**PLAY**

The Play instruction tells the program to play an animation path defined in the AMAL editor. The command is followed by a number, which tells the interpreter which animation path from the AMAL memory bank to use. On the coverdisk is an AMAL bank called 'Fly Abk'. Load this from direct mode, and view the various flight patterns using the Play instruction.

**AMAL ON, OFF, FREEZE**

These three instructions cause all AMAL paths to start, stop, or pause until started again unless a specific sprite number is included. Enter the program in Table 5 and use the keys 1 to 4 to pause and restart the spiders.

**TABLE 5**

Global a1,a2,a3,a4,x\$,a\$  
 Sprite 8,100,50,1  
 Sprite 9,140,50,1  
 Sprite 10,180,50,1  
 Sprite 11,240,50,1  
 a\$="Anim 0,(1,8)(2,8);"  
 a\$=a\$+"Loop: Move 0,100,50; Move 0,-100,50; Jump Loop;"  
 Amal 8,a\$  
 Amal 9,a\$  
 Amal 10,a\$  
 Amal 11,a\$  
 a1=0:a2=0:a3=0:a4=0  
 amal on  
 do  
 x=0  
 x\$=inkey\$  
 if x\$<>" " Then PSE  
 Loop  
 Procedure PSE  
 if x\$="1" and a1=0 then Amal Freeze 8: a1=1:  
 Goto RETURN  
 if x\$="1" and a1=1 then Amal On 8: a1=0  
 if x\$="2" and a2=0 then Amal Freeze 9: a2=1:  
 Goto RETURN  
 if x\$="2" and a2=1 then Amal On 9: a2=0  
 if x\$="3" and a3=0 then Amal Freeze 10:  
 a3=1: Goto RETURN  
 if x\$="3" and a3=1 then Amal On 10: a3=0  
 if x\$="4" and a4=0 then Amal Freeze 11:  
 a4=1: Goto RETURN  
 if x\$="4" and a4=1 then Amal On 11: a4=0  
 RETURN:  
 End Proc





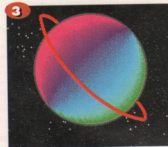
# SCREENS



Load this screen into screen 1.



Load this screen into screen 2.



Now use the dual playfield instruction to get something that looks like this! By scrolling screen 1 slightly faster than screen 2, you get a parallax effect.

**The Amiga's multiple screen modes make it a very versatile machine. However, Intuition – the software that controls the Workbench interface – is a pain to work with. AMOS screen control, on the other hand, makes light work of scrolling, flipping, animating, windows and a whole host of other functions.**

Before you can display any graphics, text or other images, a screen has to be defined and displayed. When you first run it, AMOS has already done this for you, opening a 320x200 low-resolution screen, but what happens when you want more, such as a PAL display, HAM colours or high resolution? Simple – you create a new screen!

## SCREEN OPEN, CLOSE

The Screen Open command defines a new screen and brings it to the front of the stack, making it the one currently displayed and written to. To open a screen, use the format:

**Screen Open (Screen Number), (Width), (Height), (Colours), (Resolution)**

So, to open a PAL, low-resolution screen with 64 colours, you would use the instruction:

**Screen Open 1, 320, 256, 64, Lowres**

To open the same size screen in high resolution mode with 16 colours, you would enter:

**Screen Open 1, 320, 256, 16, Hires**

To close any screen, use the Screen Close instruction. This can be followed by a number, which denotes the screen to close, or closes the current window if left without.

## SCREEN DISPLAY

With the Screen Display instruction, you can position your screen wherever you like on the monitor display, letting you create interesting

'bouncing screen' demos. The command is followed by five variables, which mark the screen number, the x position, the y position, the width of screen shown and height of screen shown respectively. See Table 1.

## TABLE 1

```
Screen Open 1, 320, 200, 32, Lowres
For c=1 to 100
  x1=rnd(300):y1=rnd(200):lnk rnd(32)
  Bar x1,y1 to x1+50,y1+50
Next c
For c=90 to 150
  Screen display 1, C,,,
  Wait Vbl
Next c
```

## SCREEN OFFSET

The Screen Offset instruction lets you do all sorts of clever scrolling. It works by displaying the current screen from a specific point – but not necessarily the top left corner. This instruction works best if you have an extra large screen, and can be used to great effect. Try the routine in Table 2.

See how easy it is to smoothly scroll a screen? In case you aren't sure, the program is displaying the screen from X-position 'X', and reads the joystick in an endless loop. As the joystick is moved, 'X' is increased or decreased, and the screen is redisplayed.

Once you've got your screens up and running, there's a lot you can do with them without actually doing much at all. These screen effects contain most of the features and functions used in commercial games, but with none of the fuss.

## DUAL PLAYFIELD

Parallax scrolling can really add something to your games, and the easiest way to create it is to use the Dual Playfield (screen one), (screen

# SCREENS

**TABLE 2**

```
Screen Open 1, 960,200,16,Lowres
For c=1 to 200
  X1=rnd(960):Y1=rnd(200):Ink rnd (15)
  Bar x1,y1 to x1+rnd(50)+1,
  y1+rnd(50)+1
Next C
x=0
Do
  Screen offset 1,x,0
  If Jleft(1) and x<1 then x=x-1
  If Jright(1) and x<640 then x=x+1
  If Fire(1) then Direct
  Wait Vbl
Loop
```

two) instruction. This takes two previously defined screens of the same resolution and overlays them, using the transparent colour (generally 0) to see through the top screen. The first screen is usually the one on top, but you can switch them around using the Dual Priority instruction. See the example given in Table 3.

**TABLE 3**

```
Screen Open 1, 540,200,16, Lowres
Screen Open 2, 640,200,16, Lowres
Screen 1
For A=1 to 501 Step 50
  Bar a,0 to a+25, 200
Next A
Screen 2
For a=1 to 601 Step 25
  Ink Rnd(16)
  Bar a,0 to a+10,20: Bar a,180 to
  a+10,200
Next A
Dual Playfield 2,1
x1=0:x2=0
Do
  Screen Offset 1,x1,0
  Screen Offset 2,x2,0
  If Jright(1) and x1<250 and x2<500
  then x1=x1+1:x2=x2+2
  If Jleft(1) and x1>0 and x2>0 then
  x1=x1-1:x2=x2-2
  If fire(1) then Direct
  Wait Vbl
Loop
```



Using the Shift Up instruction, animate this waterfall. Why not try some of your own!

**SCREEN COPY**

The Screen Copy command is the easiest way to duplicate an area of a screen and transfer it to another screen, or another area of the same screen. The format used is:

**Screen Copy (Screen Number), X1,Y1,X2,Y2 To (Screen Number), x3,y3**

X1,Y1,X2,Y2 describe the rectangular area of the screen to be copied, and X3,Y3 mark the position where the top, left corner of the block will be pasted. Load the 'Copydemo' file from the coverdisk, go to direct mode, and try these examples.

**Screen Copy 1,0,0,160,100 to 1,161,0  
Screen Copy 1,0,0,320,100 to 1,0,101**

**APPEAR**

This instruction lets you move smoothly between two pictures in a variety of ways – perfect for clearing the intro screen of a game! The instruction works by first identifying the source and destination screens, and then the effect, which can vary from one to the total number of pixels in your screen. Try the example in Table 4.

**TABLE 4**

```
Screen open 1, 320,256, 32,Lowres
Load IFF "(Your screen)",1
Screen open 2, 320,256,32,Lowres
Load IFF "(Your Screen)",2
Appear 2 to 1,81920
Wait 200
Direct
```

**FADE**

The Fade command can be used in a variety of ways. In its most basic use, it fades all the colour registers to 0 (black) at a set speed, as in:

**Fade 15**

Or you can use it to change the colour registers to a new palette, as in:

**Fade 15, \$1,\$2,\$3,\$4**

Finally you can Fade the colours to a palette taken from another screen. Load two IFF files, and enter:

**Fade 15 to 1**

If you have screen 2 displayed, this will change the palette to that of screen 1.

**SHIFT UP, DOWN**

Colour cycling can be used to great effect, as anyone who has ever messed about with *Deluxe Paint* will tell you. Shift Up moves the colours in a certain range up a step at a time through that range, and Shift Down does the opposite. The last number in the instruction tells the interpreter what to do with the end colour in the range. Try this program:

**Load IFF "Waterfall",1  
Shift Up 10,5,10,1  
Direct**

See how impressive it can be?

**PAINT THE WHOLE WORLD...**

Copper rainbows are commonly used to create complex colour backdrops to games, allowing you to have far more colours on screen than you have in your palette. The instruction is laid out like this:

**Set Rainbow number, colour, length, red, green, blue**

The number of your rainbow can be between 0 and 4. Colour is the colour index the rainbow will be based on. The length is the size of the table used to store your colour, ranging between 16 and 65500. The Red, Green and Blue indexes tell the program how to alter the basic colour index. The information for these is held in brackets, using the format (Number Of Lines, Amount to be added in a single step, Number of times to repeat the operation). See table 5.

**TABLE 5**

```
Set Rainbow
0,1,64,"(8,2,8)","(8,1,8)",""
Rainbow 0,56,1,255
Wait Key
```

Notice how the Rainbow instruction is needed to display your set Rainbow. The syntax for this instruction is:

**RAINBOW Number, first colour, vertical position, height.**

# SOUND

*How good would your favourite game be without sound effects or music? Think about it – the sound really sets the atmosphere so you'd best get familiar with AMOS's set of sonic commands.*

There are essentially two forms of sound in AMOS – samples and music. Each of these are held in their own designated memory banks and can be played across any sound channel. Samples are exactly that – raw sound that can be played at a requested rate. Music, however, is AMOS's version of a tracker module. All sounds and patterns are saved as one block, and accessed using a single command. Unfortunately, AMOS can only read AMOS music files, so you can't play your favourite tracker modules directly, but you can convert them to AMOS music files using a handy utility (see panel).

## BELL, BOOM, SHOOT

AMOS has three sounds in memory at all times – a bell, a gunshot and an explosion effect. These are played using the commands Bell, Shoot and Boom respectively. Try the example in Table 1.

### TABLE 1

For A=1 to 5  
Bell: Wait 5  
Next A  
For A=1 to 50  
Shoot: Wait 5  
Next A  
Boom: Wait 5: Boom

OK, so they may not be the most incredible effects that you have ever heard, but they'll certainly do until you start bringing in your own sounds.

## SAM PLAY

Provided you have a sample bank in memory, you can play any of the sounds within it with the Sam Play command. The command is followed by three variables, the first is the number of the sample to play, the second is the sound channel it is to be played through (0 to 3), and the third is the playback rate.

## SAM BANK

It's possible to hold more than one sample bank in memory at a time, and this command switches between them. To use it, simply type the command, followed by the number of the bank you want to switch to. If you aren't sure of the bank numbers, go to direct mode by pressing escape and type Listbank.

## SAM LOOP

This command turns all samples into looping ones. To enable it, type SAM LOOP ON. To disable it again, type SAM LOOP OFF.

## MUSIC

To play AMOS music files, you merely need to type the word Music followed by the number of the piece you want to hear. A music file can contain numerous pieces of music – one for each level of your game if you want – so including the number is vital. Without it, the command will play the first piece of music it comes across.

## MUSIC OFF

Stops all music pronto. If you have more than one piece playing, and you only want to stop one track, then use the Music Stop command.

## TEMPO

The Tempo command is used to alter the speed of any piece of AMOS music. The com-

## HOW TO MAKE A SAMPLE BANK

To make a sample bank, you need to load the 'Sample Bank Maker' program on your AMOS Program disk and run it. You'll be shown a black screen with a menu bar. Using the right mouse button, select the 'Load Sample' option, and a file requester appears.

Insert your disk of samples, and select the first one you want to include in the bank. The program will ask you for the sampling rate and then store it in memory. The sample will now be loaded and listed at the top of the screen. Repeat the process as many times as memory allows (watch the 'Memory Spare' indicator). Now just select the 'Save Bank' option from the menu, and the program does the rest.

mand is followed by a number which dictates the new tempo – the higher the number, the faster the music is played.

## MVOLUME

Mvolume is short for Music Volume, and that's precisely what it is used to set. Ranging from 0 to 63, the command changes the volume of the entire piece, not just single tracks, but used with a loop can create some useful music and sound effects. Load the demo tune (spidy.abk) on this month's coverdisk and try the listing in Table 2.

## TABLE 2

Load If "Title", 1  
Music  
Wait Key  
Fade 15  
For a=63 to 0 step -2  
Mvolume A  
wait 5  
next a

Professional looking, isn't it? That is exactly how easy it is to combine sound and graphics for stunning looking presentations, opening your AMOS world to more than just games.

## HOW TO USE A TRACKER MODULE

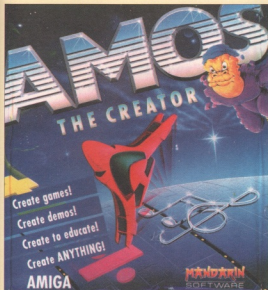
Before AMOS can play a tracker module (Soundtracker, Soundtracker etc), it needs to be converted to a format which AMOS understands – AMOS Music. This couldn't be easier if it was done for you. On your AMOS Program disk, you'll find a file called 'Soundtracker 2.1'.

Load this and run it, and a file requester will ask you for your tracker module. Insert the disk with the module on it, and select it. The rest is done for you – all you have to do is specify a file name. To load the new file, go to direct mode and type 'Load "(Filename).Abk", and the file will automatically be loaded into memory.

# SOUND

# SPECIAL OFFER

## AMOS AND AMOS COMPILER DOCUMENTATION SET



### Get the most from your free AMOS and Compiler with the official user documentation.

You've marvelled at the power and speed of AMOS and AMOS Compiler. You've tinkered with the demo programs, and maybe even created a few little routines of your own, but what now? If you really want to get the best from this stunning package, there's no substitute for the official instruction manuals.

For starters, the AMOS manual explains in simple terms how the system works. As well as this, every command is listed in detail, with clear examples and descriptions of each to get you up and running within minutes. Extensive technical appendices are also included for detailed information on the more obscure points.

Once you've got to grips with the basic AMOS language, you'll probably want to get things running even faster with the Compiler. The Compiler is available fully packaged, giving you a home for your disks and full instructions in one hit!

To complete your AMOS set, fill in the coupon below (or telephone your order on 0625 859333 quoting reference CU Amiga), indicating whether you require the AMOS manual, the Compiler manual and box, or both. The AMOS manual and Compiler set are each priced at £14.99. Cheques should be made payable to Europress Software Ltd. Alternatively, quote your Access/Visa card number, and the amount will be debited from your account.

NAME.....

ADDRESS.....

POSTCODE.....

#### PLEASE SEND ME:

- ☐ 1 AMOS manual @ £14.99
- ☐ 1 AMOS Compiler manual with box @ £14.99
- ☐ 1 each of AMOS manual and Compiler manual with box @ £24.99
- ☐ I enclose a cheque for £....., made

payable to

Europress Software Ltd.

Access/Visa card

no.....

Expiry.....

Date:.....

Please supply credit card holder's address if different from address above.

Signature:.....

Send to: Europress Software, Europa House, Adlington Park, Macclesfield, SK10 4NP. Please allow 14 days for delivery.



# AMOS COMPILER

**You know already that you're fortunate enough to have the AMOS Compiler thrown in with your free copy of AMOS, but do you know how to get the best out of it?**

For your Amiga to run an AMOS program, it has to run it first through the AMOS interpreter, which converts it into machine code, and then into the processor. This procedural takes time, which is where pure machine code programs have the edge. Or do they?

The AMOS Compiler is a handy accessory that takes your (sometimes) plodding AMOS files and converts them into pure machine code. The practical upshots of this are (a) it runs directly from disk, with no need to load the AMOS program and (b) with all conversion already done, the programs are vastly accelerated. Typically, compiled programs run at twice the speed of their BASIC counterparts.

How do you take advantage of this fabulous aid in your quest to get a game onto the shelves? The simplest way is to load the Compiler program from your main AMOS disk and compile from there, but there are other ways. The easiest is to compile from direct mode. Press escape to enter direct mode, and enter the compile command using this syntax:

**Compile "(Program name) -(disks) -(type)"**

The (disks) and (type) refer to the way the program is compiled and the type of file created. The complete list of settings is:

## DISO

- D00: Compiles from Ram Disk to Ram Disk. The fastest way to compile.
- D01: Compiles from Ram Disk to floppy disk.

## TABLE 1

Screen Open 0,320,250,64,Lowres  
For C=0 To 100

Ink  
Rnd(64):X1=Rnd(320):X2=Rnd(320):Y1=Rnd(200):  
Y2=Rnd(200) Bar (x1,y1) to (x2,y2)  
Next C

AMOS To Front

Wait Key

- D10: Compiles from disk to RAM. It's fast, but it uses a lot of memory.
- D11: Compiles from floppy to floppy. Very slow, but only holds 70K of memory.

## TYPES

- T0: Creates a Workbench friendly, stand-alone file complete with icon.
- T1: Creates a CLI-friendly program, executable from the CLI.
- T2: Not the film, a CLI program that can run in the background using Amiga multi-tasking.
- T3: Creates a compiled AMOS program that has to be run from within AMOS.

So, to compile a program completely in RAM that can run as a CLI multitasking program, you would enter:

**Compile "(Program name) -D00 -T2"**

After that, just follow the on-screen prompts.

## OTHER OPTIONS

There are a couple of other lines you can add to your Compile command, which give you more control over how the program will run when loaded independently. The first sets the default opening screen.

You'll find that AMOS compiled programs automatically open Screen 0 on loading, before running your program, which can cause a nasty flash. To get rid of this, use the extender -S0 in the instruction.

You can also choose to keep the Workbench or CLI screen intact while your AMOS program sets itself up if you wish to, which will have the effect of making everything look far more professional.

To keep the Workbench screen up, use the extender -W1. Remember to put the line 'AMOS To Front' when your program is ready to display itself. Try the example in Table 1 on the left, compiling it as:

**Compile "Test.AMOS -D01 -T2 -S0 -W1"**

## HOW TO COMPILE

For those who don't really feel like messing about with extenders and CLI-style commands, the Compiler AMOS program on your AMOS program disk provides a useful alternative. Without any programming knowledge, you can compile your programs into full machine code files faster than it takes to read this box! Here, in three easy stages, is the hassle-free compiling experience.



Load the compiler from the AMOS Program disk using the 'Load Others' option from the menu bar, and then click on 'Run Others'. Select the Compiler from the file selector and you'll be greeted with this menu screen. Here you select how the file is compiled, and what hardware is used. Along the top of the screen you'll see three icons. These represent the From, To and File selectors. Click on each a couple of times to see how you can change the from and to between Ram and floppy disk. The Type icon allows you to choose a WB compatible file, a CLI multitasking compatible file or an AMOS file. Choose the set-up that suits you, and click on the 'Compile' button.



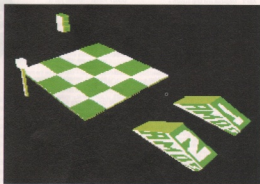
The first of two file selectors appears. Select the AMOS file you want to compile, click on 'OK', and type the name you want the compiled file to be saved under. This file doesn't need to have an AMOS extender.



Now all you can do is wait, as the yellow bar file up. After a few seconds, you'll be informed that all is done, and you can then load your compiled file and marvel at the speed.

# WHERE TO NOW?

**You think you've seen all that AMOS has to offer? You ain't seen nothing yet! This is just the beginning – your first steps into an exciting new world. Just check out what AMOS has to offer you!**



AMOS 3D lets you create anything from a business demonstration to a flight sim, using 30 new commands.

## UPDATES

AMOS 1.35 is far different from the original version. Europress have made a point of sporadically releasing update disks for the system, comprising of new commands, blonder accessories and more programming power than François Lionel ever imagined. The best thing about them, though, is that they are free! When one is available, it is instantly released to all PD libraries, not just the AMOS PDL, as well as on bulletin boards and available direct from Europress. At the moment, we're up to v1.35 – AT200 compatibility, but already we've seen improvements such as sprite flipping, full control over multi-tasking and an AMOS assembler! Roll on version 1.36!

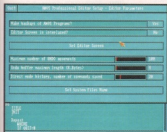


AMOS Pro is the big brother to AMOS, giving you over 750 commands and a whole host of new features. Well worth looking into.

## AMOS PRO

If the regular updates aren't enough for you, then why not fork out for a copy of AMOS Pro – this is not so much a game creator, more of a product development kit. With over 750 commands, a full debugging suite, innovative WIMP driven user interface, and with a new update disk, which is compatible with AMOS 3D and the AMOS Compiler, you can't go far wrong.

Price: £69.99 From: Europress Software. Tel: 0625 858333



These update disks can be found in any PD library, making your copy of AMOS more and more powerful.

## AMOS 3D

Any programmer will tell you that working with 3D polygon graphics can be a nightmare. Any programmer who hasn't used AMOS 3D, that is. This extension to your AMOS interpreter lets you create and manipulate 3D objects as simply as moving a sprite, and that isn't all. The 3D Object Modeller lets you build objects in a way that 3D Construction Kit could only dream of, allowing you to texture map surface detail onto the polygons, and then load them into AMOS and shift them around any way you like using 30 new commands. In BASIC, the graphics are fast enough, but compile them, and you've got speeds to rival commercial software!

Price: £34.99. From: Europress Software. Tel: 0625 858333.

## MAKE SOME MONEY!

Europress used to have a rule that any commercial software written in AMOS had to credit it, as well as display the AMOS logo within the game and on the packaging. As a result a lot of commercial publishers simply wouldn't take software written in AMOS. Not any more. Now, no mention need be made of your back-door into quality software writing. All Europress ask is that you notify them of the release beforehand and send them a copy of the finished game when it's released. Europress Software reserve the right to release the information after two months silence.

# AMOS



# TOTALLY AMOS

**When you start using AMOS, you're doing more than just using a programming language. Before you know it, you could find yourself with a new circle of friends!**



The Totally AMOS main menu. Options can be selected with the keyboard or by clicking on the numbers with the mouse.

Tutorials like this one are all well and good, but what happens when you come across a problem that you just can't solve? This booklet doesn't cover everything AMOS has to offer, and unfortunately neither does the manual. Basically, beginners can get a really hard time of it, but where can they turn for help?

To the husband and wife team of Len and Anne Tucker, that's where. These two have offered strong support for AMOS right from the very start, with Anne heading up the AMOS PD Library and Len offering technical support, as well as writing educational software such as Eurompress's Spelling Fair and Jumping Bean's Noddy's Playtime. Eighteen months ago, they put together the first issue of *Totally AMOS*, the disk magazine for the beginner.

'We saw a need for some sort of set-up to help the complete novice,' Len explains. 'We looked around at the time, and couldn't find anything that was subject specific. Everything

seemed to assume that people knew what a For Next loop was, or what a While Wend was. We set up *Totally AMOS* to help people who needed it. Write to us, and we'll do a tutorial on it, that sort of thing. Another aim behind *Totally AMOS* was to set up connections between programmers and artists, musicians and so on. Both things were what we saw was needed, and we tried to create this environment – something like a beginners' club. What we really want is for members to feed off each other's knowledge.'

For the record, the entire thing was Anne's idea, and consequently she does most of the writing in terms of putting the magazine together. Len is mainly responsible for the magazine driver, which is being continually enhanced. But before I go any further, let's take a look at the product itself.

*Totally AMOS* works from an interactive menu and displays text pages and illustrations at your command – a cross between Multimedia and teletext in that sense. Everything is controlled from the mouse or numeric keys, so there's no confusion from the start. But that isn't going to sell it.

What will, though, is the editorial content. Broken into 10 main sections, each broken down further, the disk contains reviews of AMOS PD and AMOS support titles, comprehensive news and letters pages, a debating corner – where readers can slag each other and the editorial team off as much as they want – and, of course, the help pages.

Help comes in two different ways. The first is from a Question and Answer session, where the smallest and simplest problems are solved. Other help comes in the form of com-

plete tutorials, covering all aspects of a problem. Subjects covered in tutorials in past issues include AMOS 3D, How To Get The Most Out Of The AMOS Compiler, AMAL, and a guide to the AMOS commands not mentioned in the manual.

One other feature is a spotlight on leading AMOS programmers – those who have created the most impressive public domain software and routines. If there's no other reason to work hard at your coding, the promise of an interview feature must be enough to entice most to submit work.

At the moment, *Totally AMOS* sells around 150 copies, but that looks set to change thanks to a new distribution deal that will see the magazine on sale in Canada and the USA, and then Australia.

*Totally AMOS* costs £2.50 per issue, and back issues cost £3.00 each. If you subscribe, you become eligible for a 10 per cent discount on all disks from the AMOS PDL. If you want to try it out, there is a PD issue available from the AMOS PDL for £2.00. Whatever space on the disk isn't taken up with the magazine is filled with useful routines and programs, making it a serious bargain. For more information, contact Len or Anne on (0792) 588156.

## AMOS IN ACTION

As well as running the AMOS PDL, writing *Totally AMOS* and various educational packages, Len and Anne are also responsible for a new AMOS book simply titled *AMOS in Action*. The book is essentially a guide on how to write an arcade game in AMOS Basic, taking you from the sort of set-up you really need to write a game, right up to completing the project. A disk is included, containing a complete version of the demonstration game in question – Marvin The Martian – and costs a mere £12.95. If you buy it from the AMOS PD Library, it costs an even merer £10.35. For more information, call Len or Anne on (0792) 588156.



# TOTALLY AMOS

## TOTALLY AMOS



Just one of the many Help pages, to get you out of those sticky situations.



# AMOS PUBLIC DOMAIN

**So what exactly can be done with AMOS? One place to start looking is in the various public domain libraries, where dozens of disks contain-  
ing AMOS programs and routines can be found.**

## CARD GAMES 2 game

Everyone has those moments when you want to do nothing other than sit down with a pack of cards, and deal a quick hand of Patience. Or at least David Lerner seems to think so, or he wouldn't have come up with *Card Games 2*, a collection of nine different Patience variants. Being on your own needn't be a chore any more!

In case you're not the kind of person who enjoys spending long evenings alone, *Patience* is a card game for one. Generally it involves a number of card stacks, which have to be rearranged using a series of set rules to reach a certain position – four rows of ascending cards of individual suits, for example.

On the whole the game isn't particularly taxing – it depends on the luck of the deal more than anything else – but it does while away the time.

All nine games are accessed from a single menu screen, and to be honest there isn't a great deal of difference between them. Each game is displayed on the same blank backdrop with the same set of cards and an identical control method involving two clicks with the left mouse button – one to pick up a

card and another to put it down again. The presentation isn't much, but it's such a great version that I haven't any time left to write this article!

**Disk No:** APD448. **From:** AMOS PDL, 1 Penrynnydd Road, Swansea, SA5 7EH. **Tel:** 0792 588156. **Price:** £2.00. **Compatibility:** All machines. **Memory:** 512K

76%

## FOOTBALL/SPEEDY REEDY game

*Football* is possibly the most pointless management game around, and that's what's so great about it. Take something like *Tracksuit Manager*, remove all traces of management so that all you have left is the results screen, and you've got *Football*. It sounds like a strange idea, but you do find yourself clicking through

the screens just to see who wins the league. No playability or gameplay, but fun.

*Speedy Reedy*, however, is playable. Playable and a lot of fun. In this *PacMan*-style maze game, the aim is to eat the power pills as they appear while staying out of the clutches of the evil ghost. It's all rather unfair, as the ghost can float through walls and you can't, but help is at hand. Collect a speed-up, and you can race all over the shop without fear of being caught. Superb samples and music really make the game stand out – they just have to be heard to be believed.

**Disk No:** APD462. **From:** AMOS PDL, 1 Penrynnydd Road, Swansea, SA5 7EH. **Tel:** 0792 588156. **Price:** £2.00. **Compatibility:** All machines. **Memory:** 512K

72%

## U-TILITIES utilities

Just to show that AMOS isn't solely for making games, Tony Swanwick's dynamic duo of applications far outclass a lot of the more commercial PD utilities around. The first program, *U-File*, is a fully comprehensive file editor, allowing you to load individual files from disk and tailor them to your own desires. If you would rather have your name than Mike Singleton's on the title screen of *Midwinter 2*, then this is the gizmo to do it with. To be used carefully.

The other program, *U-Zone*, is an AMOS help application that lets you define screen zones for an IFF graphic. Anyone who has created a menu screen, and then spent hours trying to get the positioning of the buttons pixel perfect will know how much of a struggle it can be. This package is dedicated to you. Thanks to a few icons and some very well-written code, you'll be able to define every screen zone as easily as drawing a box on *DeluxePaint*. A must for all AMOS owners.

**Disk no:** APD454. **From:** AMOS PDL, 1 Penrynnydd Road, Penlan, Swansea, SA5 7EH. **Tel:** 0792 588156. **Price:** £2.00. **Compatibility:** All machines. **Memory:** 512K.

91%



For those moments when you have nothing to do – AMOS *Patience*!



It might look like *PacMan*, but that's where the resemblance ends with *Speedy Reedy*.

*U-Zone* is a godsend for people who like using buttons, but don't like making them!



# AMOS

1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299

1323  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299

Gen

AM  
£37.5

AM  
57  
A  
A  
Ac

Lo

Amstr  
Amstr  
Cass  
Cass  
Coma  
Egon  
Egon  
Egon  
Egon  
NEC  
Panas  
Panas

Rin

054

054



A bit late for Valentine's Day, but fast nonetheless. A shot from Digital Orgasm.

## DIGITAL ORGASM demo

Despite the somewhat dubious title this demo, which comes from coders Cubic, is suitable for all the family and serves to prove that AMOS is just as good at creating light sourced vector objects and dot flags as most other programming utilities are.

The nicest thing about it is that it's the latest in a long line of demos flying back and forth in a friendly competition between Cubic and coding rivals Fanatix, and they just get better and better. Every time one team comes up with something new, the other has to reply by doing the same thing, only bigger, better and faster.

If you are a collector of megademos, then you probably won't find much to impress you here. After all, machine code routines will always be faster than AMOS, but if you take into account that these routines are written in a version of BASIC, you can't help but be impressed.

All in all, though, this isn't a very entertaining demo. I would have liked to have seen a lot more visual effects and a lot less text on the screen, or even some more entertaining text. Still, it shows what AMOS is capable of so far.

Disk no: APD456. From: AMOS PDL, 1 Penrynnydd Road, Penlan, Swansea, SA5 7EH. Tel: 0792 588156 Price: £2.00. Compatibility: All machines. Memory: 512K.

71%



Turbo Text is an excellent word processor – and a snip at £2.00!

## TURBO TEXT text editor

The mysteriously named Harbinder Ghag is responsible for this handy AMOS word processor. The screen layout is more or less the same as most others (ruler bar at the top of the screen, most options selected from a menu bar) but that's where most of the similarity ends.

All the usual options are included, such as loading and saving ASCII files (which makes this perfect for writing your AMOS routines on), and various formatting controls – which are, incidentally, perfectly arranged. They do exactly what you would expect, unlike many PD word processors, which seem to have more than a few unpredictable results. On top of these are a few options not normally seen. For a start, you can set the word processor to read your text as you type. Unfortunately it uses the Amiga speech synthesis which everyone knows is about as decipherable as the old Spectrum Currah Speech unit, but it works well in allowing you to keep your eyes off the screen if you should so desire. You can also get the program to read the entire document back to you, which gives you an excellent way of looking over what you've written if you don't like reading your own work.

Disk no: GPD145. From: AMOS PDL, 1 Penrynnydd Road, Penlan, Swansea, SA5 7EH. Tel: 0792 588156 Price: £2.00. Compatibility: A500, A500+, A600. Memory: 512K.

92%



A must for any guitar player, Tab Master will solve those Tabulature blues and let you get back to playing them instead.

## TAB MASTER/HECTIC 2/DEARY DIARY miscellaneous

There are three cool programs written by one David Meager included on this disk. He may be only 14 years of age, but he's already creating professional looking software! Tab Master is a must for any guitar owner, allowing you to enter musical notation on a stave, which is then converted into guitar tablature instantly and, if you want, is marked out on a fretboard. No more messing around with mnemonics for me. Next time I want to transcribe Mendelssohn, I'll just use this.

Alongside it on the disk is Hectic 2, an interesting tile-based puzzle game which involves picking up numbered tiles to get the highest score possible. Some tiles take points off your score, and some add to it.

This might seem a little on the easy side, but when you add to that the fact that the first player can only move the cursor horizontally on the board and the second player can only move it vertically, you realise that you've actually got yourself a real challenge.

The diary program, which is simply titled Dear Diary, is really nothing to write home about I'm afraid, but when put on a disk with two great programs like these, you can't really complain.

Disk no: GPD180. From: AMOS PDL, 1 Penrynnydd Road, Penlan, Swansea, SA5 7EH. Tel: 0792 588156 Price: £2.00. Compatibility: All machines. Memory: 512K.

87%

PUBLIC DOMAIN

# AMOS USER GROUPS

*Being an AMOS user can boost your social life! This is the claim we make based on the sheer number of AMOS User groups there are in Europe alone. Here's a complete list of who to write to.*

## AMOS User Club UK

Aaron Fothergill  
1 Lower Moor  
Whiddon Valley  
Barnstaple  
North Devon  
EX32 8NW

## AMOS Programmer's Exchange

7 Majestic Road  
Hatch Warren  
Basingstoke  
Hampshire  
RG22 4XD

## Klub AMOS France

BP 133  
18003 Bourges Cedex,  
France

## Tom Poulsen

Danish AMOS Group  
Stenmollen 28  
2640 Hedehusene  
Denmark

## AMOS Club Nederland

Kerkeind 8a  
5293 AB Gemonde (NB)  
Holland

## Belgium Club

Johan Francois  
Wilgenpark 7  
9900 EEKLO  
Belgium

## AMOS Club USA

Mark H. Budziszewski & Mark A.  
Shultz  
PO Box 11434  
Milw.  
WIS 53211,  
USA

## AMOS NTSC Club

David Lazarek  
516 E 11th Street  
Michigan City  
IN 46360,  
USA

## Aaron Wald

201-19 Tonnele Avenue  
Jersey City  
NJ 07306,  
USA

## Deutsche

Carsten Bernhard  
Asterweg 4  
6229 Walluf  
Germany

## Portugal

Eduardo David  
Rua Nina Marques Pereira N 92 - Esq  
1500 Lisboa  
Portugal

## WIN A STACK OF DISKS

How do you fancy putting your new found knowledge to the test? How would you like to win 30 - yes THIRTY disks of your choice from the AMOS PD Library? Like the sound of that? Here's what you have to do. On the coverdisk is a program called 'Spidy AMOS'. This is a very basic program based on a Mac program called *Neko*. At the moment, all that you do is tease a spider with the mouse pointer, trying to keep it out of its grasp while at the same time keeping it interested enough to chase. What we want you to do is soup it up.

Yes, you have the basic program, now see what you can do with it! Maybe the Spider should carry a rocket launcher? Maybe the screen should scroll? What do you think? You have complete freedom to do whatever you like. The best entry wins, it's as simple as that. So what are you waiting for! Get it together, and stick your entry on a disk and pop it off to us at CU with a covering letter explaining the changes made. Remember to mark your envelope 'Magnetic Media - Do Not Xray'. Send your disks to: I WANT ALL THAT LOVELY PD, CU Amiga, Priory Court, 30-32 Farringdon Lane, London EC1R 3AU. Closing date is 30th June. The editor's decision is final and no correspondence will be entered into. Employees of EMAP Images or the AMOS PD Library are not allowed to enter, although we can't figure out who from the AMOS PD Library would want to. After all, they've already got the disks, haven't they?

# USER GROUPS





# TICK, CUT & POST

**SYSTEC PD (CU) 2 RIDGE ROAD,  
LEITCHWORTH, HERTS, SG6 1PH**

## TELEPHONE

**(0462) 483604**

# System pd

## DISK PRICES

1-10 £1.25 EACH  
11-20 £1.15 EACH  
21+ £1.00 EACH

LOOK FOR THE CUB LOGO



### ASSASSINS' GAMES+

These are the best value packs of PD games you'll ever find!

The Assassins' disks 1-40 contain nearly 200 of the best PD games available. This pack is a must for any serious player.

DISKS 1-10...£9.95 □

DISKS 11-20...£9.95 □

DISKS 21-30...£9.95 □

DISKS 31-40...£9.95 □

DISKS 1-40...£37.95 □

### PRECISION-BRANDED DISKS

25¢ 14.95 35¢ £23.95  
50¢ 29.95 65¢ £39.95  
Price includes delivery.

### CUB ART PACK

This 7 disk set of mostly 32-bit routines everything from fonts, binary loaders and helixes, to quantisers, trees and transport. There are hundreds of programs and routines to be loaded into your favourite art or CPU package.

□ 7 DISK CUBART SET £6.95

### CALL THE SYSTEC BBS

Lot of free files to download!

(0462) 483604

6.00pm to 9.00pm  
seven days a week

MAXIMUM SPEED OF 2400 BAUD

### AM/PM

The brilliant disk magazine for anyone into music on their Amiga.

Each issue is packed with tips on using popular music software, along with Amiga and MIDI music files.

AM/PM Magazine £3.45

AM/PM Sample Disk £3.45

ISSUE 10 OUT NOW

### 150 FONTS

This 3 disk pack contains over 150 fonts with different and many other art and CPU packages.

1 Disk...£3.95

### ONLY £1.00

TIME TO GET DOWN SAMPLES

Probably the largest collection of Amiga samples ever. Each disk comes with instruments, melodies, and much more.

1 Disk...£1.00

ONLY £1 EACH □

### A1200 AGADISK 1

A collection of programs and routines for the Amiga A1200 and A1200. The disk contains working AGA graphics, fonts and other routines.

AGADISK 1...£1

### WE STOCK THE FULL RANGE OF FISH AND TBAID DISKS

### ANIMATIONS

34800 Shaping Steel

34710 Agony 3d

34640 Amiga

34510 Bubble Cool

34410 Bounce

34310 Call the Wind

34210 Chain Reaction

34110 Chase

34010 Chase

33910 Chase

33810 Chase

33710 Chase

33610 Chase

33510 Chase

33410 Chase

33310 Chase

33210 Chase

33110 Chase

33010 Chase

32910 Chase

32810 Chase

32710 Chase

32610 Chase

32510 Chase

32410 Chase

32310 Chase

32210 Chase

32110 Chase

32010 Chase

31910 Chase

31810 Chase

31710 Chase

31610 Chase

31510 Chase

31410 Chase

31310 Chase

31210 Chase

31110 Chase

31010 Chase

30910 Chase

30810 Chase

30710 Chase

30610 Chase

30510 Chase

30410 Chase

30310 Chase

30210 Chase

30110 Chase

30010 Chase

29910 Chase

29810 Chase

29710 Chase

29610 Chase

29510 Chase

29410 Chase

29310 Chase

29210 Chase

29110 Chase

29010 Chase

28910 Chase

28810 Chase

28710 Chase

28610 Chase

28510 Chase

28410 Chase

28310 Chase

28210 Chase

28110 Chase

28010 Chase

27910 Chase

27810 Chase

27710 Chase

27610 Chase

27510 Chase

27410 Chase

27310 Chase

27210 Chase

27110 Chase

27010 Chase

26910 Chase

26810 Chase

26710 Chase

26610 Chase

26510 Chase

26410 Chase

26310 Chase

26210 Chase

26110 Chase

26010 Chase

25910 Chase

25810 Chase

25710 Chase

25610 Chase

25510 Chase

25410 Chase

25310 Chase

25210 Chase

25110 Chase

25010 Chase

24910 Chase

24810 Chase

24710 Chase

24610 Chase

24510 Chase

24410 Chase

24310 Chase

24210 Chase

24110 Chase

24010 Chase

23910 Chase

23810 Chase

23710 Chase

23610 Chase

23510 Chase

23410 Chase

23310 Chase

23210 Chase

23110 Chase

23010 Chase

22910 Chase

22810 Chase

22710 Chase

22610 Chase

22510 Chase

22410 Chase

22310 Chase

22210 Chase

22110 Chase

22010 Chase

21910 Chase

21810 Chase

21710 Chase

21610 Chase

21510 Chase

21410 Chase

21310 Chase

21210 Chase

21110 Chase

21010 Chase

20910 Chase

20810 Chase

20710 Chase

20610 Chase

20510 Chase

20410 Chase

20310 Chase

20210 Chase

20110 Chase

20010 Chase

19910 Chase

19810 Chase

19710 Chase

19610 Chase

19510 Chase

19410 Chase

19310 Chase

19210 Chase

19110 Chase

19010 Chase

18910 Chase

18810 Chase

18710 Chase

18610 Chase

18510 Chase

18410 Chase

18310 Chase

18210 Chase

18110 Chase

18010 Chase

17910 Chase

17810 Chase

17710 Chase

17610 Chase

17510 Chase

17410 Chase

17310 Chase

17210 Chase

17110 Chase

17010 Chase

16910 Chase

16810 Chase

16710 Chase

16610 Chase

16510 Chase

16410 Chase

16310 Chase

16210 Chase

16110 Chase

16010 Chase

15910 Chase

15810 Chase

15710 Chase

15610 Chase

15510 Chase

15410 Chase

15310 Chase

15210 Chase

15110 Chase

15010 Chase

14910 Chase

14810 Chase

14710 Chase

14610 Chase

14510 Chase

14410 Chase

14310 Chase

14210 Chase

14110 Chase

14010 Chase

13910 Chase

13810 Chase

13710 Chase

13610 Chase

13510 Chase

13410 Chase

13310 Chase

13210 Chase

13110 Chase

13010 Chase

12910 Chase

12810 Chase

12710 Chase

12610 Chase

12510 Chase

12410 Chase

12310 Chase

12210 Chase

12110 Chase